AL-TP-1992-0035-Vol-3

FUNCTIONAL REQUIREMENTS OF AN ADVANCED INSTRUCTIONAL DESIGN ADVISOR: SIMULATION AUTHORING (VOLUME 3 OF 3)

M. David Merrill
Mark K. Jones

Department of Instructional Technology
Utah State University
Logan, UT 84322

Douglas M. Towne

Behavioral Technology Laboratories
University of Southern California
Los Angeles, CA 90089

Earl R. Nason

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
Brooks Air Force Base, TX 78235-5000

September 1992

Interim Technical Paper for Period March 1990 - February 1991

AIR FORCE MATERIEL COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5000

NOTICES

This technical paper is published as received and has not been edited by the technical editing staff of the Armstrong Laboratory.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

EARL R. NASON
Project Scientist

HENDRICK W. RUCK, Technical Director
Technical Training Research Division

RODGER D. BALLENTINE, Colonel, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1992	Interim – March 1990 – February 1991

4. TITLE AND SUBTITLE

Functional Requirements of an Advanced Instructional Design Advisor: Simulation Authoring (Volume 3 of 3)

5. FUNDING NUMBERS

C – F33615-88-C-0003
PE – 62205F
PR – 1121
TA – 10
WU – 43

6. AUTHOR(S)

M. David Merrill
Mark K. Jones
Douglas M. Towne
Earl R. Nason

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Department of Instructional Technology
Utah State University
Logan, UT 84322

Behavioral Technology Laboratories
University of Southern California
Los Angeles, CA 90089

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

Armstrong Laboratory
Human Resources Directorate
Technical Training Research Division
Brooks Air Force Base, TX 78235-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL-TP-1992-0035-Vol-3

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Dr. Daniel J. Muraida, (512) 536-2981

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Advanced Instructional Design Advisor (AIDA) will provide automated and intelligent assistance to inexperienced instructional computer-based instructional developers. This technical paper contains two reports discussing simulation capability and AIDA. First, Drs M. David Merrill and Mark K. Jones describe the integration of transaction shells (TAX) with RAPIDS, a simulation-based instructional authoring and delivery system. They argue RAPIDS could be used to build simulations and some portions of the knowledge representation model required by AIDA. Merrill and Jones present an example of the instructional design process using the AIDA transaction shells. Then, Dr Douglas Towne discusses the instructional development process using the RAPIDS system. Towne describes specifying course structure, building device simulations, producing simulations and delivering instruction using RAPIDS. He walks through the RAPIDS instructional development process using instruction on the maintenance of the U.S. Air Force T-38 jet aircraft as an example. Two significant conclusions resulted from this work. First, incorporating RAPIDS into the AIDA experimental prototype (AIDA) would be too costly and time-consuming. Second, because AIDA does require some simulation creation capability, a simpler simulation authoring capability will be included in the near term version of the system.

14. SUBJECT TERMS

Advanced Instructional Design Advisor
Computer-based instruction
Computer-based training
RAPIDS
Simulation
Transaction shells

15. NUMBER OF PAGES
122

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

PREFACE

The work reported herein was done for the Advanced Instructional Design Advisor project at the Air Force Armstrong Laboratory (Human Resources Directorate). The substance of this research was done under contract to Mei Associates, Inc., the primary contractor on the Advanced Instructional Design Advisor (Contract No. F33615-88-C-0003).

This work was done as part of the second phase effort on the Advanced Instructional Design Advisor. The initial phase of this project established the conceptual framework and functional specifications for the Advanced Instructional Design Advisor, an automated and intelligent collection of tools to assist subject matter experts who have no special training in instructional technology in the design and development of effective compute. · based instructional materials. This second phase provided the design specifications for an experimental prototype.

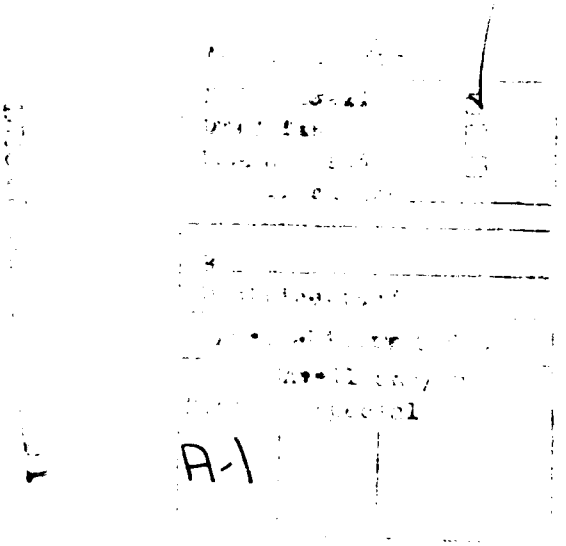
Mei Associates' final report for the second phase is being published as an Armstrong Laboratory Technical Paper (Hickey et al., 1991). In addition, Mei Associates received nine papers from various consultants working on this phase of the project. These nine papers have been grouped into 3 sets and edited by AL/HRTC personnel. They are published as Volumes 1 - 3 of Functional Requirements of an Advanced Instructional Design Advisor:

Volume 1: Epitomizing Functions

Volume 2: Task Analysis and Troubleshooting

Volume 3: Simulation Authoring

This is Volume 3 in the series. Earl Nason wrote Sections I and IV. Drs. David Merrill and Mark Jones wrote Section II. Dr. Doug Towne wrote Section III.



SUMMARY

The Advanced Instructional Design Advisor is an R & D project being conducted by the Air Force Human Resources Laboratory in response to an Air Training Command (ATC) Manpower, Personnel, and Training Need calling for improved guidelines for authoring computer-based instruction (CBI) (MPTN 89-14T).

Aggravating the expensive and time-consuming process of CBI development is the lack of Air Force personnel who are well-trained in the areas of instructional technology and educational psychology. More often than not, a subject matter expert with little knowledge of CBI is given the task of designing and developing a computer-based course. Instructional strategies that work in a classroom are often inappropriate in a computer-based setting (e.g., leading questions may work well in a classroom but are difficult to handle in a computer setting). Likewise, the computer offers the capability to present instruction in ways that are not possible in the classroom (e.g., computer simulations models can be used to enhance CBI).

The Advanced Instructional Design Advisor is a project aimed at providing subject matter experts who have no background in computer-based instructional systems with automated and intelligent assistance in the design and development of CBI. The goal is to reduce CBI development time while insuring that the instructional materials are effective.

I. INTRODUCTION (NASON)

The Advanced Instructional Design Advisor is an R & D project aimed at providing automated and intelligent assistance to inexperienced instructional designers who have the task of designing and developing computer-based instruction (CBI). The particular problem being addressed by this line of research is the need for more cost efficient methodologies for the design and development of CBI. Current methods for developing CBI are expensive, time-consuming, and often result in ineffective instruction due to the general lack of expertise in computer-based instructional systems (Spector, 1990).

The Advanced Instructional Design Advisor project is divided into four phases.

Phase 1: Conceptualization & Functional Specifications

Phase 2: Conceptual Refinement & System Specifications

Phase 3: Prototype, Field Test, & Refinement

Phase 4: Technology Demonstration & System Validation

The first two phases have been performed under Task Order Contracts. The third phase is being accomplished via a Broad Agency Announcement (BAA). The fourth phase will be funded by a fully specified contract. The work reported here concerns the second phase for which Mei Associates' was the primary contractor.

Mei Associates' final report for the second phase is being published as an Armstrong Laboratory Technical Paper (Hickey et al., 1991). In addition, Mei Associates received nine papers from various consultants working on this phase of the project. These nine papers have been grouped into 3 sets and edited by AL/HRTC personnel. They are published as Volumes 1 - 3 of Functional Requirements of an Advanced Instructional Design Advisor:

Volume 1: Epitomizing Functions

Volume 2: Task Analysis and Troubleshooting

Volume 3: Simulation Authoring

This is Volume 3 in the series.

During the second phase, the educational consultants to the Advanced Instructional Design Advisor project, as well as developing system specifications, considered including simulation

capability. Including simulation capability in the Advanced Instructional Design Advisor was desirable from an educational standpoint. It, also, seemed possible because of the availability of RAPIDS, a simulation-based instructional system. RAPIDS, which provides robust computer-based simulation authoring and delivery capabilities was developed for the Air Force Armstrong Laboratory by the University of Southern California.

The next section of this technical paper is a report Drs. M. David Merrill and Mark K. Jones of M. David Merrill Consulting, prepared for Mei Associates under the Advanced Instructional Design Advisor contract. This section describes the integration of transaction shells (TRX), the basis of the Advanced Instructional Design Advisor, with RAPIDS. Merrill and Jones argue RAPIDS could be used not only to build simulations but also to build some portions of the knowledge representation required by the Advanced Instructional Design Advisor. The Advanced Instructional Design Advisor could, then, then automatically develop the instruction using the knowledge representation and incorporate the simulations developed by RAPIDS. Merrill and Jones present an example of the instructional design process using the Advanced Instructional Design Advisor transaction shells. They also discuss how the shells and RAPIDS could be interfaced.

In the third section of this technical paper, Dr. Douglas Towne, from the University of Southern California Behavioral Technology Laboratories, discusses the instructional development process using the RAPIDS system. Towne divides his discussion of RAPIDS into four parts: specifying course structure, building device simulations, producing simulations and delivering instruction. He walks through the RAPIDS instructional development process using instruction on the maintenance of the U.S. Air Force T-38 jet aircraft as an example.

The final section of this technical paper briefly summarizes the above reports and discusses the advantages of including simulation capability in the Advanced Instructional Design Advisor. Finally, the feasibility of integrating RAPIDS' simulation capabilities and the Advanced Instructional Design Advisor is described.

II. RAPIDS AND TRANSACTION SHELLS (MERRILL & JONES)

This section describes the integration of Transaction Shells (TRX) with a simulation-based instructional system (RAPIDS). The section combines details of the instructional design process with a walkthrough of an example based on maintenance troubleshooting of the starting process for the T-38 dual engine jet aircraft.

This section is divided into four steps:

knowledge analysis,
strategy analysis,
transaction authoring and
instructional delivery.

While presented in order, recognize that the steps are actually performed iteratively, in a spiral fashion, with important linkages and sharing of data between the steps.

The knowledge representation model described in this report has previously been published in:

Jones, M.K. Li, Z. & Merrill, M.D. (1990). Domain knowledge representation for instructional analysis. Educational Technology, 30 (10) 7-32. Used by permission.

The enterprise authoring algorithm, the classes of transaction shells, and the list of example transaction parameters, were previously published in:

Jones, M.K. Merrill, M.D., & Li, Z. (1990). Implementation of an expert system for instructional design: Design of the strategy analysis component. McLean, VA: Human Technology Inc.

Knowledge Analysis

Knowledge analysis is the step whereby knowledge of the domain to be instructed (in this case, aircraft maintenance procedures) is elicited from subject matter experts and represented in a domain knowledge base which may be used by all other steps and tools in the course development process. For this purpose we have developed a domain-independent knowledge representation model.

The Knowledge Representation Model

Knowledge is represented by objects we call frames; each frame has an internal structure, and external links to other frames. These external links are termed elaborations of the frame. A set of elaborated frames linked together, containing

all the knowledge required for instruction leading to acquisition of an integrated human performance, or enterprise, is called an elaborated frame network.

There are three kinds of frames:

entities corresponding to some thing, for example a device, object, person, creature, place, or symbol;

activities, or groups of related actions to be performed by the learner and

processes or groups of related actions entirely external to the learner.

There are four kinds of elaborations. These are:

attributes, which represent characteristics of a frame;

components, which represent constituents of a frame. For an entity, the components would be parts of the entity; for an activity, steps; and for a process, events and causes;

abstractions, which correspond to a "kinds-of" class/sub-class hierarchy into which the frame may be classified; and

associations, which are links to other frames in the network.

The network structure of the knowledge representation allows information to move through the structure, so data contained in one part of the net affect the data stored elsewhere. Two principal means by which this occurs are:

inheritance, in which attributes of a class or super-class in an abstraction hierarchy are passed to a sub-class or instance and

propagation, in which the contents of a frame influence the contents of another frame connected to it via an association link.

The knowledge representation model focuses on identifying aspects of knowledge which may serve as a basis for making instructional decisions.

Entity Frames

Entities are things in the real or imagined world including objects (natural objects and manufactured devices), creatures (animals and persons), places (natural and constructed), and

symbols. Examples of entities include the Eiffel Tower, a tractor and George Washington.

We assume that there can be no instruction without at least one entity involved. The only enterprise that may be performed with only an entity is denoting.

Activity Frames

An activity is some group of actions performed, or which could be performed, by the learner. Examples of activities include operating a device, using a formula to perform a calculation, and participating in a social interaction, such as group decision-making.

Process Frames

A process is some group of actions outside the learner including physical and social events in the real or imagined world. Examples of processes include the functioning of a device, the transmission of a disease, desertification, cell replication, planetary motion, group decision-making, and evolution.

The distinction between an activity and a process has to do with the role of the learner. If the learner will, or could be an actor, then the actions are analyzed as an activity, with the learner's potential role central to the analysis and instruction. If, on the other hand, the learner could not be an actor and the actions occur entirely external to the learner, then the actions are analyzed as a process. For example, the procedure to service a machine, which will be performed by the learner, is an activity; replicating a cell in biology is a process.

Relations

In addition to the frame, the other fundamental structure in instructional design is the relation. Relations are structures that link and attach meaning to a set of frames. Each frame links to the relation, and from the relation to other frames in the set, in a manner specified by the relation. The semantics of the particular relation give meaning to the individual links.

Elaboration, and Elaborated Frame Networks

Frames, as previously defined, have an external structure of links to other frames. These are termed elaborations of the frame. The identification of relations (links) to a frame we call elaborating a frame.

A set of linked elaborated frames is termed an elaborated frame network, or EFN. A single EFN corresponds to the knowledge

elements and interrelations required to support performance of an enterprise. A course may include instruction to facilitate acquisition of one, or several, enterprises, thus the domain knowledge base for a course may contain one or a set of EFNs. A set of EFNs is itself an EFN.

There are four kinds of elaborations:

attributes, which represent characteristics of a frame;

components, which are the constituents of a frame;

abstractions, which represents the generality of frames and

associations, which are non-hierarchical aggregations of frames.

Attributes

An attribute is a labelled set of values from which objects and their properties may take values over time. Attributes define the characteristics of frames. The constraints placed on a particular attribute determine the legal values that may be taken by objects having that attribute. For example, an attribute labelled "color" defined for a frame "Apple" may have legal values "red," "green," "yellow." The set {red, green, yellow} associated with the label "color" and the constraint that only one value from the set may be applied to any single object, together define this attribute.

The following operators may be used to define legal values for attributes: the booleans AND, OR, XOR, NOT; the logicals =, <>, <, <=, >, >=; and the range operator .. .

Attributes take values from the set of legal values defined for the attribute. The value selected during analysis is termed the initial value. In addition, storage may also be reserved for a current value.

The Component Elaboration

Each kind of frame has its unique component structure. Entities have parts, activities have steps, processes have both events, and causes.

Attributes and components are similar in structure. The distinction has to do with independence of existence. A component has independent status, and therefore, can be independently inserted and deleted; an attribute is an essential part of an entity's definition. For example, a "computer" frame might be defined as the aggregate of the attributes "manufacturer", "model", "ID #"; and also as the aggregate of the

parts "monitor", "circuit boards", "disk drive", "keyboard". A computer without a monitor still retains its identity as a computer (one that lacks a monitor). On the other hand, a computer without a manufacturer violates the definitional character of computer (at least for the schema defined for that knowledge base).

Entity Components. An entity can be described by its parts. Each part has at least a name, and an associated function. If the entity is a physical object, then each of its parts may also have a location and a graphical description. A part, of course, can have sub-parts.

Activity Components. Learning to perform activities is central to maintenance training. Examples of maintenance activities include repair procedures, diagnostic and troubleshooting procedures, and applying safety rules.

An activity consists of one or a series of steps. Steps are performed in a specified sequence, including loops and conditions. Each step has a set of actions associated. These actions may be performed in a specified sequence (algorithm), including loops and conditions, or they may be triggered by events (heuristics). All actions are represented in the analysis as action + trigger(s) + consequence(s). Triggers and consequences are either sequence data, changes in attributes, time values, or a combination

Each activity must link to at least one entity which is an actor (performs action and can vary its behavior in the activity based on some internalized knowledge). A step, or an action, of an activity may itself be an activity, with its own sub-steps.

Process Components. The understanding of processes plays an important part in maintenance activities, for example in troubleshooting mechanical, electrical, and electronic devices.

The constituents of a process are its stages; each stage has an associated event topology. An event may itself be a process, or in this context, a sub-process.

An event is a transformation of inputs to outputs. Inputs are either attributes, actions, or time values. Outputs are either attributes, actions, time values, or stage transitions. Transformations are either mathematical, logical, or both. An event may encapsulated within an entity, or be abstract.

Events may be linked together into event topologies. A topology is like a network, except that it need not be fully interconnected. The network structure supports dependencies among events (the output of event A is the input to event B), feedback (the output of event B becomes an input back into event

A), and self-regulation. The addition of timing signals as inputs and outputs (these may be relative or absolute) supports synchronization and temporal dependencies. An event topology describes how a process behaves in respect to changes in its inputs.

A stage is sequentiable phase within a process which has an event topology that is distinct from those of other stages. So, a process behaves fundamentally differently in respect to changes in its inputs from one stage to another. A process need have only one stage. Examples of processes with stages are an electronic device, with stages Off, Powerup, and Operating; and reproduction in a seed, with stages Dormant and Active.

The Abstraction Elaboration

Abstraction represents the generality of entities, activities, and processes. The levels of abstraction are instance and class, with relationships among classes in a multi-level generalization hierarchy represented by sub-class and super-class links. The abstraction elaboration is exactly equivalent to generalization.

An instance represents a specific entity such as a particular object, person, symbol, or place. It may also represent a particular activity, or a particular process. The instance represents the lowest level of abstraction and has no subordinates.

A class is an abstract frame that represents general features held in common by two or more frames. Attributes, and their legal values, help define the class. Instances in the class share these attributes.

The Association Elaboration

Associations are non-hierarchical aggregations of frames. Unlike the other aggregation elaborations (attributes and components) in which one frame is viewed as the aggregate of the others in the relation, for an association *each* frame in the relation may be thought of as an aggregate of the other frames. An example of an association is a process linked with another process. Each process may be used as an analogy for the other. The relation is the same seen from the point of view of either process.

Classes of Associative Relations. There are many relations among things in the world. Only those relations that have instructional value are included in the knowledge representation. These relations either provide meaningful information for prescribing instruction, or combine with other knowledge structures and relations to promote acquisition of an enterprise.

Collections

Collections are sets of frames all of the same class. For example, if the class "passenger" were defined, as well as several instances of passenger, such as "J. Smith," then a collection of passengers could be defined that would include a group of passenger instances. This collection could be labelled "passengers_flight_75."

Collections, unlike the other relations, are not considered to be elaborations. This distinction, though somewhat arbitrary, is based on the collection being able to be substituted for frames in certain situations. Collections, however, do not have the same status as frames; rather it is the objects of which a collection is formed that are frames. In semantic data model terms, a collection is a grouping relation.

Collections should not be confused with classes, in that they are not more abstract than the frames which form the collection. Hence, there can be no inheritance from a collection to its members. Generalization and inheritance are defined on the class structure of the underlying frame, not the collection. Collections, however, may have attributes associated with them. These non-inheritable attributes perform functions such as summarizing across the collection. For example, "passengers_flight_75" might have an attribute "count."

The relationship between a collection and its underlying frames is extended to allow the definition of collections of collections. A sub-collection of "passengers_flight_75" might be "1st_Class_passengers_flight_75."

Collections may take the place of frames in associations. For example, it may be most appropriate to represent that an activity frames "uses" a collection of entity frames, rather than a single entity.

Collections may also take the place of frames as attributes, and as components. For example, the parts of a "computer" entity may include a collection of "memory chips."

A single frame, of course, may be a member of more than one collection. Collection membership is defined by an expression using a syntax equivalent to that for defining class membership.

Integrating RAPIDS

Knowledge analysis, the process of representing subject matter using the knowledge representation model, is relatively straightforward. Knowledge acquisition, the process of eliciting the subject matter from an SME in such a form that it can be represented using the knowledge representation model, is more difficult.

The challenge is to find methods of eliciting knowledge from the subject matter experts that are intuitive and couched in terminology and conceptual structures relevant to the domain, but are sufficiently structured to allow automatic translation to the formalisms of the KR model.

RAPIDS provide such a method for the identification of entity and process components, in the steps of building the device simulation. The production of objects identifies entities and their components, the specification of behavior of objects in effect identifies the events topologies of processes.

This part of RAPIDS could be extended so it generates parts of the EFN knowledge structure automatically as a result of building the device simulation. This would then be augmented by other methods to acquire other elements of the subject matter, such as activity components, and abstraction.

Example Knowledge Analysis for T-38 Maintenance Training

This example is for the engine starting procedure.

Entities

left engine starting circuitry
right engine starting circuitry
left start button
right start button
left timer
right timer
left ignitor
right ignitor
left engine
right engine
external power
left generator
right generator
left generator light
right generator light
diverter valve
air
fuel
ignition spark

left ac bus
right ac bus
battery
throttle
front panel
left engine instruments
right engine instruments
left transformer rectifier
right transformer rectifier
static inverter

Activities

start left engine
start right engine
on ground start
emergency start, no engines running
in air start, 1 engine running

Processes

engine ignition
generation of ac power
diverter valve operation
cross over system

Attributes

diverter valve
    state: (centered, left, right)
    air_in, air_out_left, air_out_right (numeric)

actuator
    force (numeric)

Entity Components

left engine starting circuitry
    left start button
    left timer
    left ignitor
    left generator
    left generator light
    left ac bus
    left transformer rectifier

right engine starting circuitry
    right start button
    right timer
    right ignitor
    right generator
    right generator light

```
          right ac bus
          right transformer rectifier

Activity Components

     start engine (right or left)
          1 check fore, aft, and under aircraft
          2 apply external air
          3 depress engine start button 14% rpm
          4 advance throttle to idle
          5 if fuel flow >= 360 lb/hr and no ignition, retard
               throttle to off; wait 2 minutes; go to 3
          6 check engine instruments
          7 check hydraulic pressure
          8 check caution light panel

Process Components

     diverter valve operation
          inputs: (left_actuator.force, right_actuator.force,
               diverter_valve.air_in);
          outputs: (diverter_valve.air_out_right,
               diverter_valve.air_out_left);

     transformation:
          (if left_actuator.force >= right_actuator.force then
               diverter_valve.air_out_left <-
               diverter_valve.air_in;
               diverter_valve.air_out_right <- 0
          else if right_actuator.force >= left_actuator.force
               then diverter_valve.air_out_right <-
               diverter_valve.air_in; diverter_valve.air_out_left
               <- 0
          else diverter_valve.air_out_left <- 1/2
               diverter_valve.air_in;
               diverter_valve.air_out_right <- 1/2
               diverter_valve.air_in)

Abstractions

     engine starting procedures
          on ground start
          emergency start
          in air start

     engine
          left engine
          right engine

     engine starting circuitry
          left engine starting circuitry
          right engine starting circuitry
```

starting button
    left starting button
    right starting button

timer
    left timer
    right timer

ignitor
    left ignitor
    right ignitor

generator light
    left generator light
    right generator light

ac bus
    left ac bus
    right ac bus

engine instruments
    left engine instruments
    right engine instruments

transformer rectifier
    left transformer rectifier
    right transformer rectifier

Associations

process diverter_valve_operation *involves* entities
    left_actuator, right_actuator

activity start_engine *uses* entities air, throttle, start button, fuel, engine_instruments

activity start_engine *applies* process engine_ignition

Transaction Authoring

TRX provides a library of reusable instructional programs, or transaction shells, for the delivery of instruction. These programs contain generalized instructional algorithms, each appropriate for teaching a certain type of content, but do not contain any content. Each shell incorporates several parameters, configurable by the author, which control the functioning of the shell during course delivery.

An instructional *transaction* is a particular instructional interaction with a student. A transaction is characterized as a bounded interchange between an instructional system and a

student, which facilitates the acquisition by the student of a specified competence. Transactions comprise the entire range of instructional interactions including: one-way transmission of information (e.g. video, lecture, or document -- which are not very good transactions because they lack interaction); discussions and conversations; tutoring (e.g. traditional CAI and Intelligent Tutoring Systems); simulations; and micro-worlds (with or without coaching).

The effectiveness of a transaction is determined by the extent of the relevant active mental processing required and the nature of the learner's interaction with the content to be learned. An adequate transaction can assume both expository and inquisitory modes; it allows the degree of learner or system control to be adjusted; it includes display and response parameters which allow the transaction to customized for different learners, different subject matters and different delivery systems. Each transaction is also capable of being invoked to perform different instructional functions with its content: overview; familiarity instruction; basic instruction; example; practice; remediation; and assessment.

A *transaction shell* is a piece of computer code which when executed causes a given transaction to take place. A transaction shell knows what knowledge it must have to execute its interaction with the learner. It can query the domain knowledge base to find the required knowledge and thus be able to instantiate its knowledge slots. If the domain knowledge base does not contain the necessary knowledge, the transaction shell can direct the user/designer to supply the required content.

Once a transaction has been selected or prescribed, it must then be configured and authored. Configuration involves setting the parameters, modifying the strategy, and attaching the content. Authoring involves attaching domain specific instructional materials to the instructional structure set up by the transaction. For example, in concept learning by compare/contrast, the transaction would contain all the elements to generate examples, practice, and assessment items for concept learning. However, it would require that images of the different concepts, with the defining attributes indicated, be provided by the designer. Each transaction shell knows what domain specific data it requires, and will guide the designer in preparing and entering that data.

Each transaction shell has default values for each of its parameters, including its strategy elements. While most practical instruction will require changing these parameters, the acceptance of default values allows the rapid prototyping of instruction incorporating the content and instructional strategies identified. This rapid prototyping allows the designer to get a feel for the structure and look of the finished

course while still early enough in the design process to easily change design decisions. The effects of making different design choices can also be easily compared.

Transaction shells reside in a *transaction library*. In addition, configured and authored components are also stored in the library. The library supports the reuse of components, and is a key element in improving the efficiency of the design process.

Classes of Transaction Shells

There are several classes of transactions, with each class differentiated from the others by their knowledge structures and performance components instructed. The primary transaction classes are component, abstraction, and association. Component transactions instruct all or part of one component hierarchy (parts, steps, or events) in the elaborated frame network. Abstraction transactions instruct all or part of an abstraction hierarchy. Association transactions instruct two or more frames linked by an association relation. Enterprise transactions require as a knowledge base all frames and their interrelations for a given enterprise (Enterprise transactions will be discussed further in the section on the Strategy Analysis).

Within each primary class are a number of subclasses. There are 12 subclasses: for components, the subclasses of identify, execute, and interpret; for abstraction, the subclasses of judge, classify/decide, generalize, and transfer; for association, propagate, analogize, and substitute.

Component Transactions

There are three classes of component transactions corresponding to the three types of knowledge frames: identify for entity frames, execute for activity frames, and interpret for process frames.

An identify transaction requires either an instance or class entity frame. It enables the student to acquire the names, functions, properties, and relative location of all the parts which comprise an entity. The student knows *what* it is.
An execute transaction requires either an instance or class activity frame. It enables the student to acquire the steps of the activity. The student knows *how* and is able to do the activity.

An interpret transaction requires either an instance or class process frame. It enables the student to acquire the events and causes in a process. This means that student knows *why* it works and can explain the events which lead to a given

consequence or can predict the consequence from a series of events.

Abstraction Transactions

Different types of abstraction transactions can be discriminated on the basis of the performance required and the different combinations of frames from an abstraction hierarchy involved in the transaction. We have identified at least four classes of abstraction transactions: judge, classify/decide, generalize, and transfer.

A judge transaction requires a class frame with two or more subordinate instance frames. These frames can be entity, activity, or process frames. It enables the student to acquire the ability to order the instances of a given class on the basis of some dimension (criterion). The dimensions can be any attribute or combination of attributes. Judging the performance of others as they perform an activity is an example. Ordering a set of objects is an example.

A classify/decide transaction requires a superclass frame with two or more subordinate class frames each of which have two or more instance frames. These frames can be entity, activity, or process frames. It enables the student to acquire the ability to sort or classify instances as to class membership. It enables the student to know *when* to select one alternative from another. Concept identification is an example. Deciding among alternative activities to accomplish some goal is an example. Editing (selecting the appropriate usage) is an example.

A generalize transaction requires a superclass frame with two or more subordinate class frames each of which have two or more instance frames. These frames can be entity, activity, or process frames. Generalization transactions enable the student to acquire the ability to combine instances of two or more classes into a more general class. Generalization is the inverse of classification.

A transfer transaction requires a superclass frame and one or more class frames. These frames can be entity, activity or process frames. It enable the student to acquire an abstraction model, that is, a generalized set of steps for an activity , or a generalized set of events for a process, and to apply this abstraction model to a previously unencountered class or instance of the activity or process.

Association Transactions

Different types of association transactions can be discriminated on the basis of the performance required and the different combinations of frames from a set of associated frames

involved in the transaction. We have identified at least five classes of association transactions: propagate, analogize, substitute, design, and discover.

A propagate transaction requires two or more associated frames. The most common relations between knowledge frames, uses, requires, applies, all involve propagation. A propagation transaction makes a deliberate effort to facilitate the student's integration of information from two or more associated knowledge frames. One of the most important propagation associations is the link between an application activity and a tool activity; another is the link between a method activity and a process. Propagation enables the student to acquire one set of skills in the context of another set of skills. While learning an application activity, the student can simultaneously learn the tool activity for doing the application. While learning a tool, the student can simultaneously learn application activities for the tool. While learning a process, the student can simultaneously learn a method activity for studying or observing the process. While learning a method activity, the student can simultaneously learn the process for which the method was devised.

An analogize transaction requires two or more knowledge frames linked by the relation *analogy for*. It enables the student to acquire the steps from one activity by likening it to a similar activity; or to acquire the events in one process by likening it to a similar process or activity.

A substitute transaction requires two or more knowledge frames linked by the relation *alternative for*. It enables the student to learn an alternative activity or process by comparison, elaboration, or extension of a previously learned activity or process. It also enables the student to acquire alternative ways to accomplish a given activity or to explain a given process.

Transaction Shells for T-38 Maintenance Training

An instructional transaction approach to curriculum development enables us to think of the knowledge and skills to be learned at a more integrated level. Rather than thinking of individual skills it is desirable to identify complex sets of related activities and to build the curriculum around the acquisition of these complex human enterprises. The interactions necessary to promote the acquisition of all of the knowledge and skill associated with a given enterprise make up a transaction family.

Henry Halff suggested a "rough list of the tasks that a maintainer must master in order to effectively maintain a piece of equipment." We suggest that this list provides a first cut

list of the kinds of human enterprises that an Air Force maintenance curriculum might enable a learner to acquire. Halff's list includes the following enterprise classes: equipment operation, equipment calibration and adjustment, equipment testing, access and disassembly, equipment repair, and troubleshooting. The curriculum consists of several specific instances of each of these classes. A more detailed analysis of the curriculum will identify each of these specific enterprises and the knowledge and skill which makes up each of these enterprises.

The type and sequence of interactions necessary to acquire each of these complex enterprises is different. It is proposed that a high level transaction manager be designed and developed for each of the different types of curriculum enterprises. This transaction manager would be a program that could be easily configured to call and sequence the primary transactions identified as necessary for this curriculum. We would propose the development of transaction families to support the enterprise classes of the maintenance curriculum. Halff also suggested that acquiring a model of the equipment was a necessary component of any maintenance training. In addition to transaction families for each of the 6 maintenance training enterprises identified we also suggest an equipment model family of transactions which will be a component of each of the other transaction families. In addition he suggested as a possible procedure "redesign and jury rig." This is a complex enterprise for which we have also defined a separate transaction family.

Figure 1 identifies the 5 transaction shell instances which we believe are necessary to enable learners to acquire the knowledge required to construct a mental model of a particular device or piece of equipment. This particular transaction family does not represent a "stand-alone" enterprise but is a necessary component of the transaction family required for every other maintenance training enterprise. Two classes of transactions are represented: identify - (1) physical & conceptual structure; and interpret - (2) device functioning, (3) device configuration, (4) fault recognition, and (5) prediction.

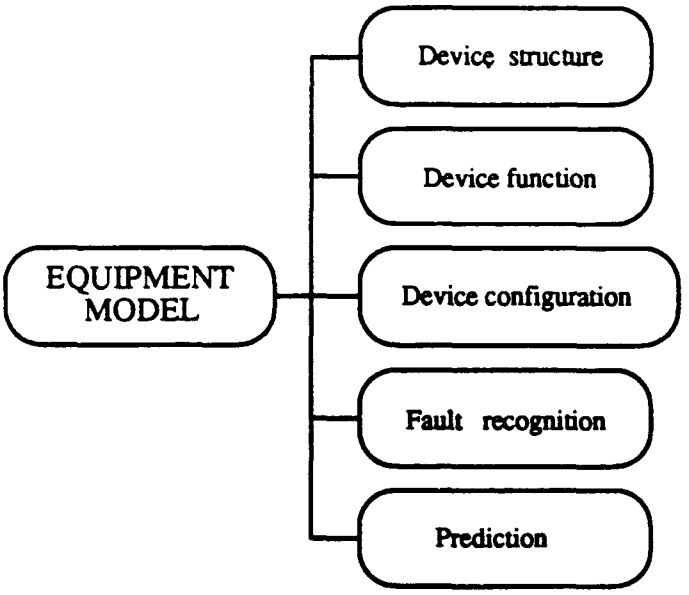


Figure 1. Transaction family for acquiring an equipment mental model

Figure 2 identifies the 2 transaction shell instances and the two nested transaction families which we believe are necessary to enable learners to acquire the knowledge and skill required to operate a particular device or piece of equipment. Two classes of transactions are represented: execute - (6) equipment operation procedures; and decide/classify - (7) operation procedure or job aid selection.

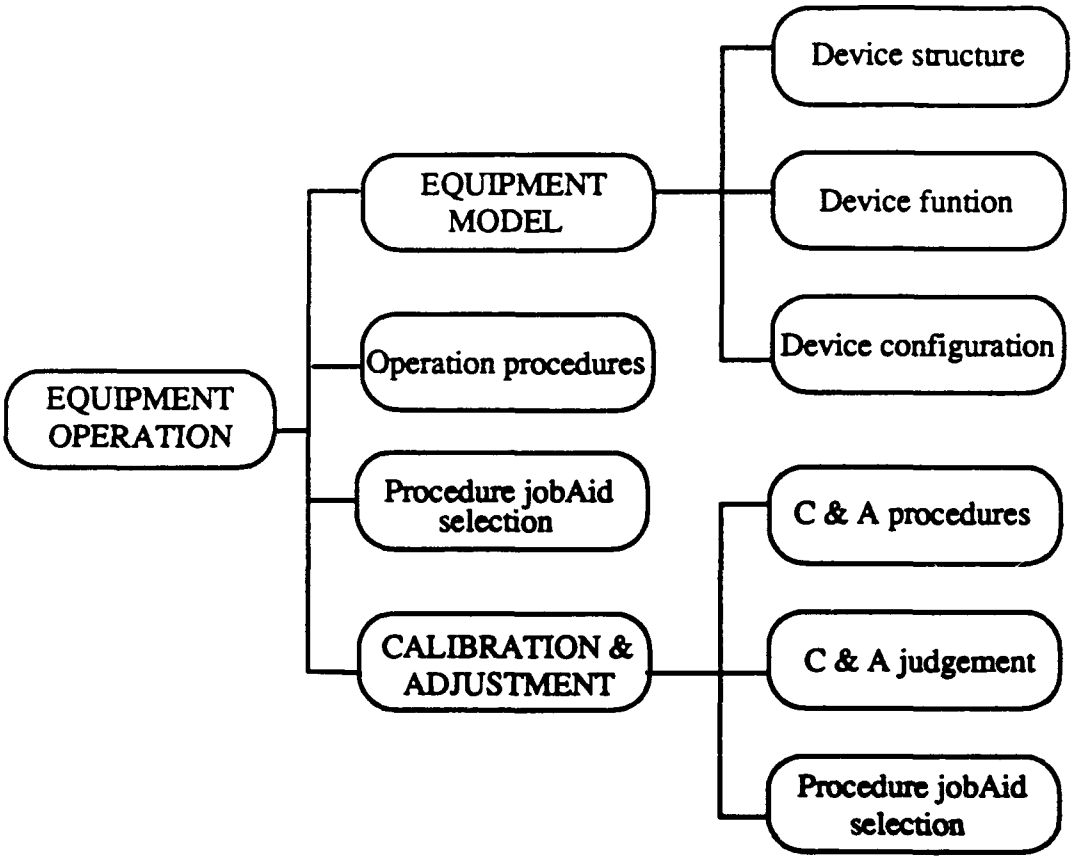


Figure 2. Transaction family for acquiring equipment operation enterprises

Figure 3 identifies the 3 transaction shell instances and the nested transaction family which we believe are necessary to enable learners to acquire the knowledge and skill required to calibrate and adjust a particular device or piece of equipment. Three classes of transactions are represented: execute - (8) calibration and adjustment procedures; judge - (9) calibration and adjustment judgement; and decide/classify - (10) calibrate and adjust procedure or job aid selection.

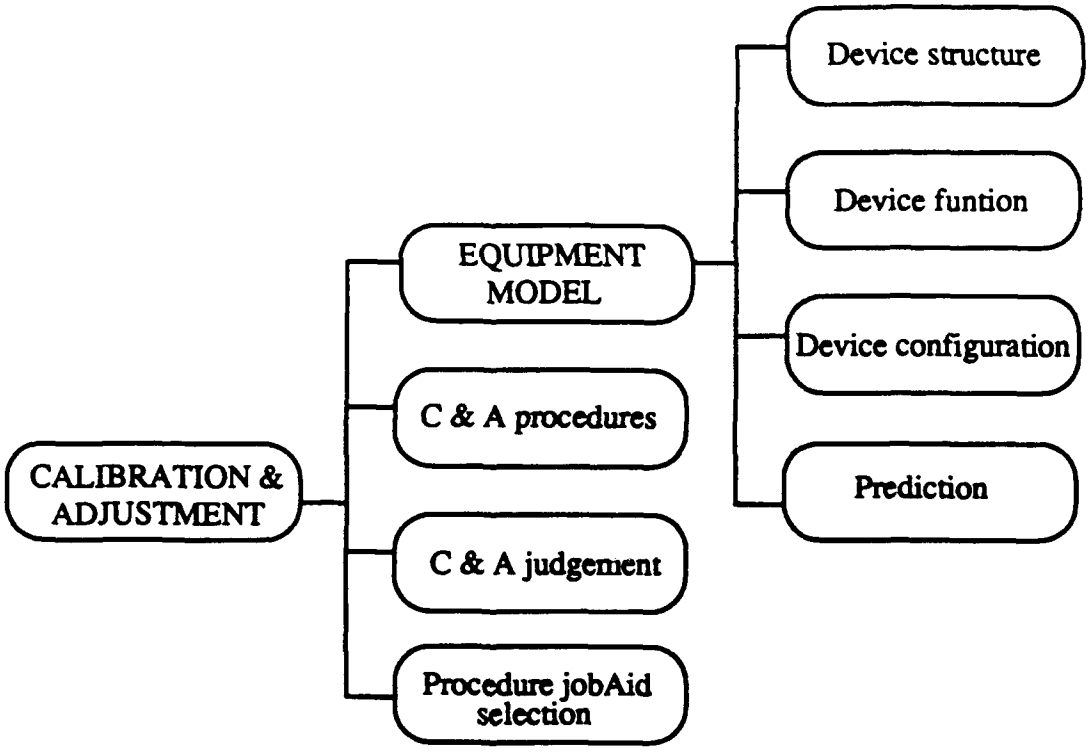


Figure 3. Transaction family for acquiring equipment calibration and adjustment enterprises

Figure 4 identifies the 3 transaction shell instances and the nested transaction family which we believe are necessary to enable learners to acquire the knowledge and skill required to test a particular device or piece of equipment. Three classes of transactions are represented: execute - (11) testing procedures; judge - (12) testing judgement; and decide/classify - (13) test procedure or job aid selection.

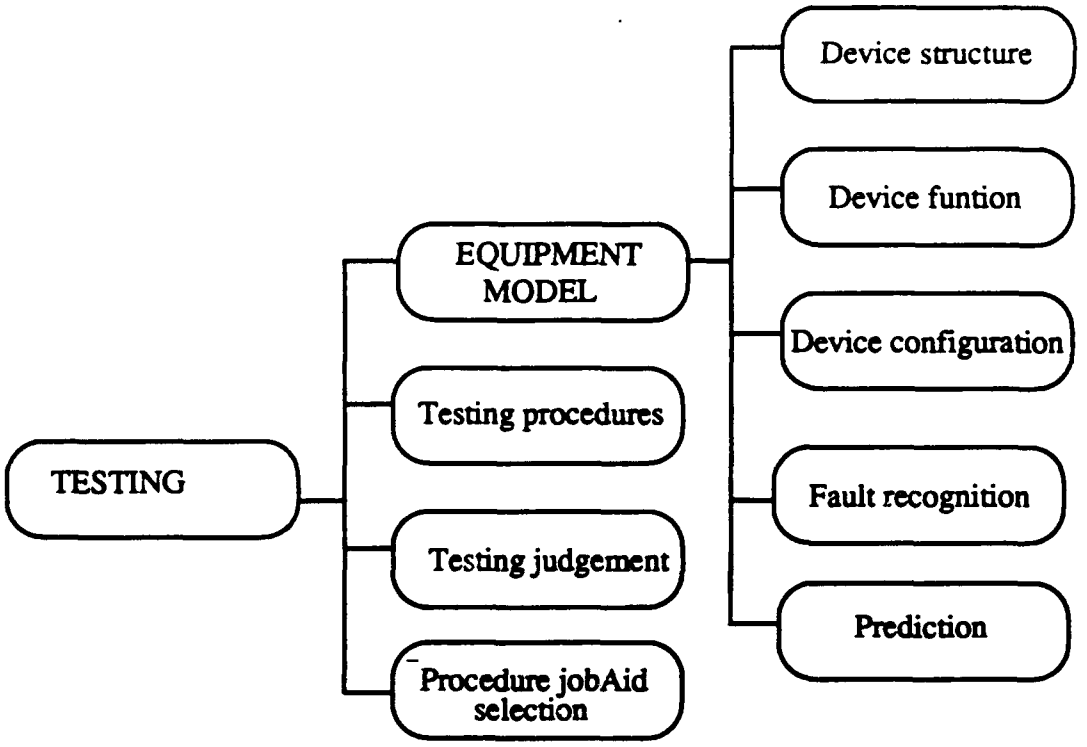


Figure 4. Transaction family for acquiring equipment testing enterprises

Figure 5 identifies the 2 transaction shell instances and the nested transaction family which we believe are necessary to enable learners to acquire the knowledge and skill required to access and disassemble a particular device or piece of equipment. Two classes of transactions are represented: execute - (14) access and disassembly procedures; and decide/classify - (15) access and disassembly procedure or job aid selection.

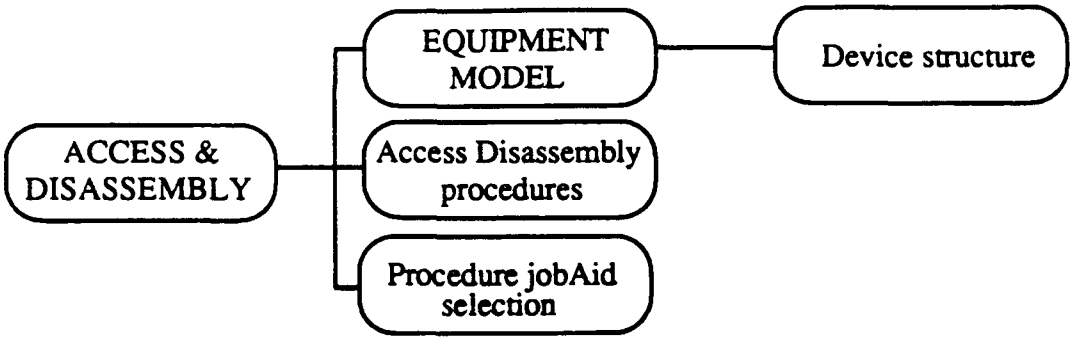


Figure 5. Transaction family for acquiring equipment access and disassembly enterprises

Figure 6 identifies the 2 transaction shell instances and the 4 nested transaction families which we believe are necessary to enable learners to acquire the knowledge and skill required to repair a particular device or piece of equipment. Two classes of transactions are represented: execute - (16) repair procedures; and decide/classify - (17) repair procedure or job aid selection.

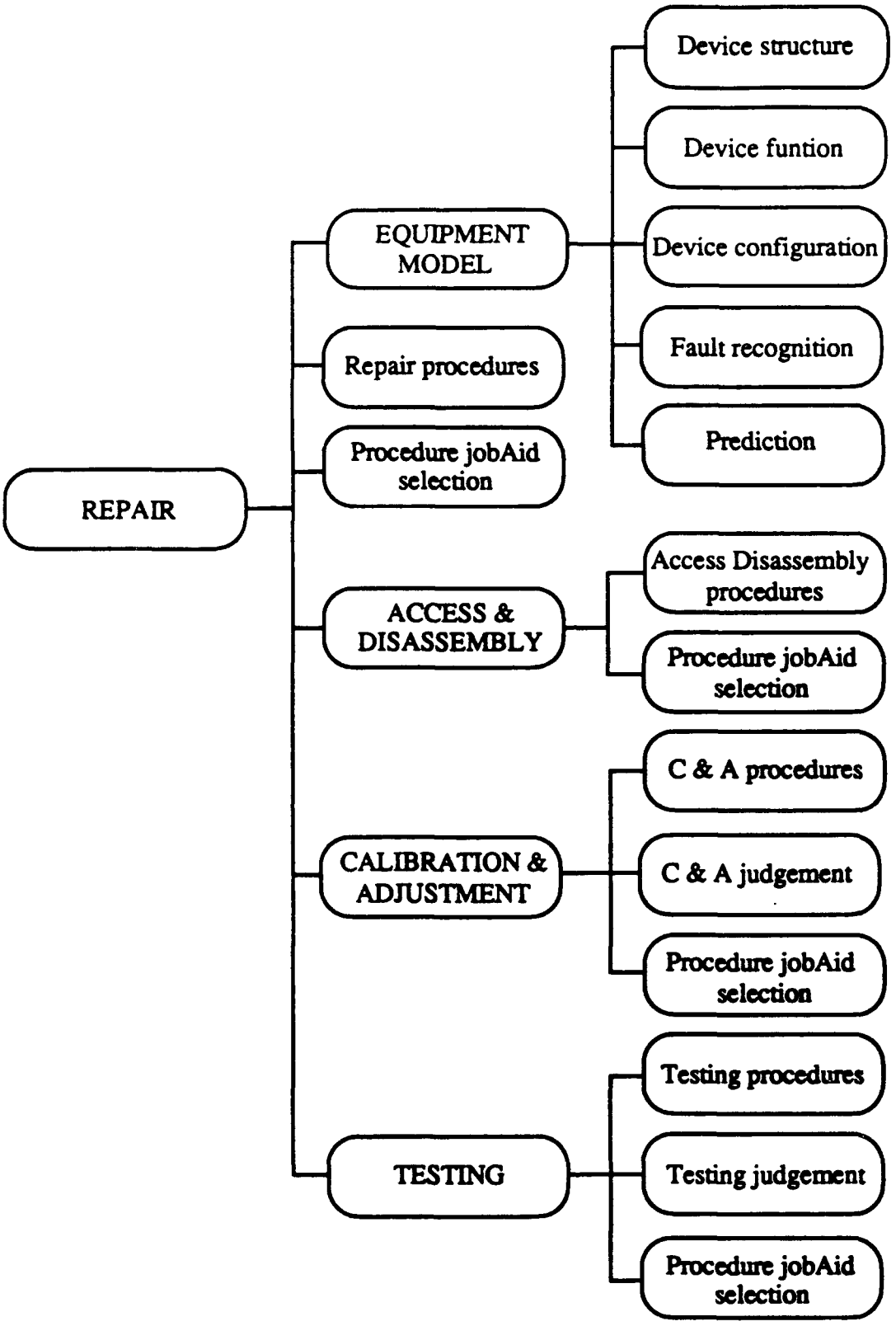


Figure 6. Transaction family for acquiring equipment repair enterprises

Figure 7 identifies the 4 transaction shell instances and the 4 nested transaction families which we believe are necessary to enable learners to acquire the knowledge and skill required to trouble shoot a particular device or piece of equipment. Two classes of transactions are represented: execute - (18) logical fault isolation procedures, and (19) intuitive fault isolation procedures; and decide/classify - (20) logical fault isolation procedure or job aid selection and (21) intuitive fault isolation procedure or job aid selection.

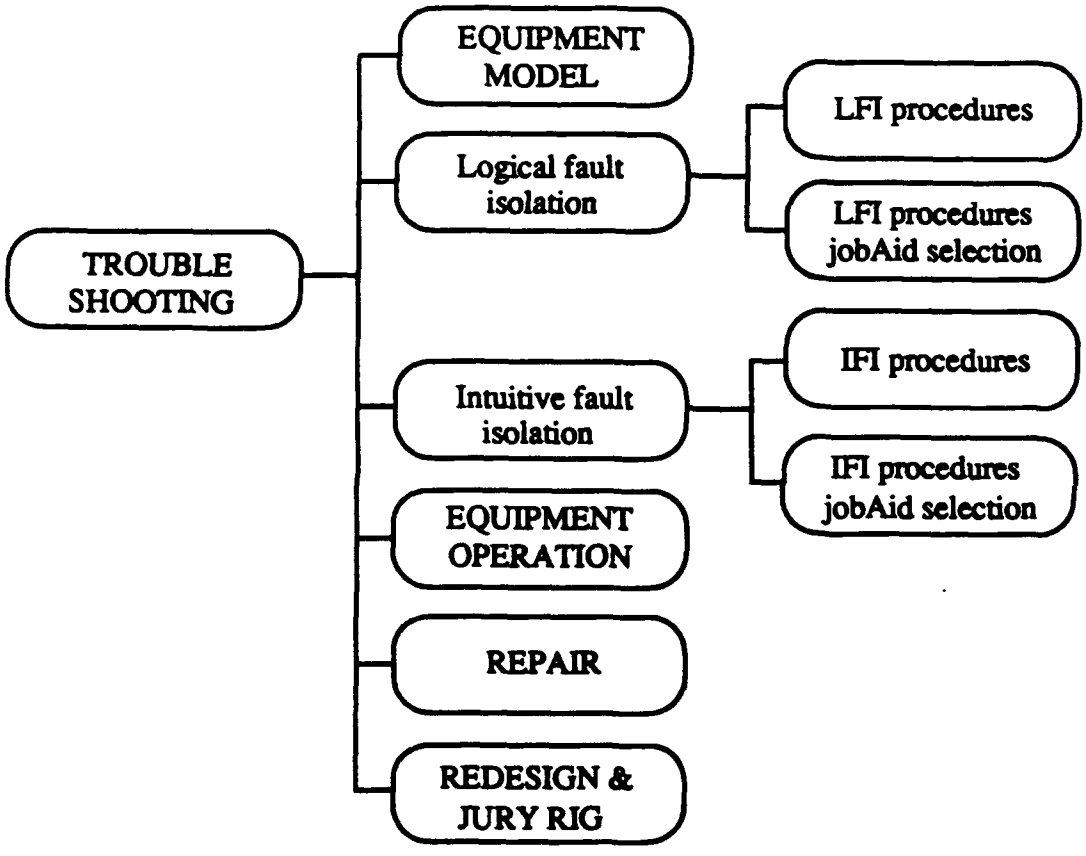


Figure 7. Transaction family for acquiring equipment trouble shooting enterprises

Figure 8 identifies the 6 transaction shell instances and the 3 nested transaction families which we believe are necessary to enable learners to acquire the knowledge and skill required to

redesign or jury rig a particular device or piece of equipment. 5 classes of transactions are represented: execute - (22) heuristic jury rig procedures; decide/classify - (23) redesign or jury rig procedure selection; transfer - and (24) procedural transfer; analogize - (25) conceptual and procedural analogies; substitute - (26) conceptual, functional, and procedural substitution; and design - (27) redesign techniques.

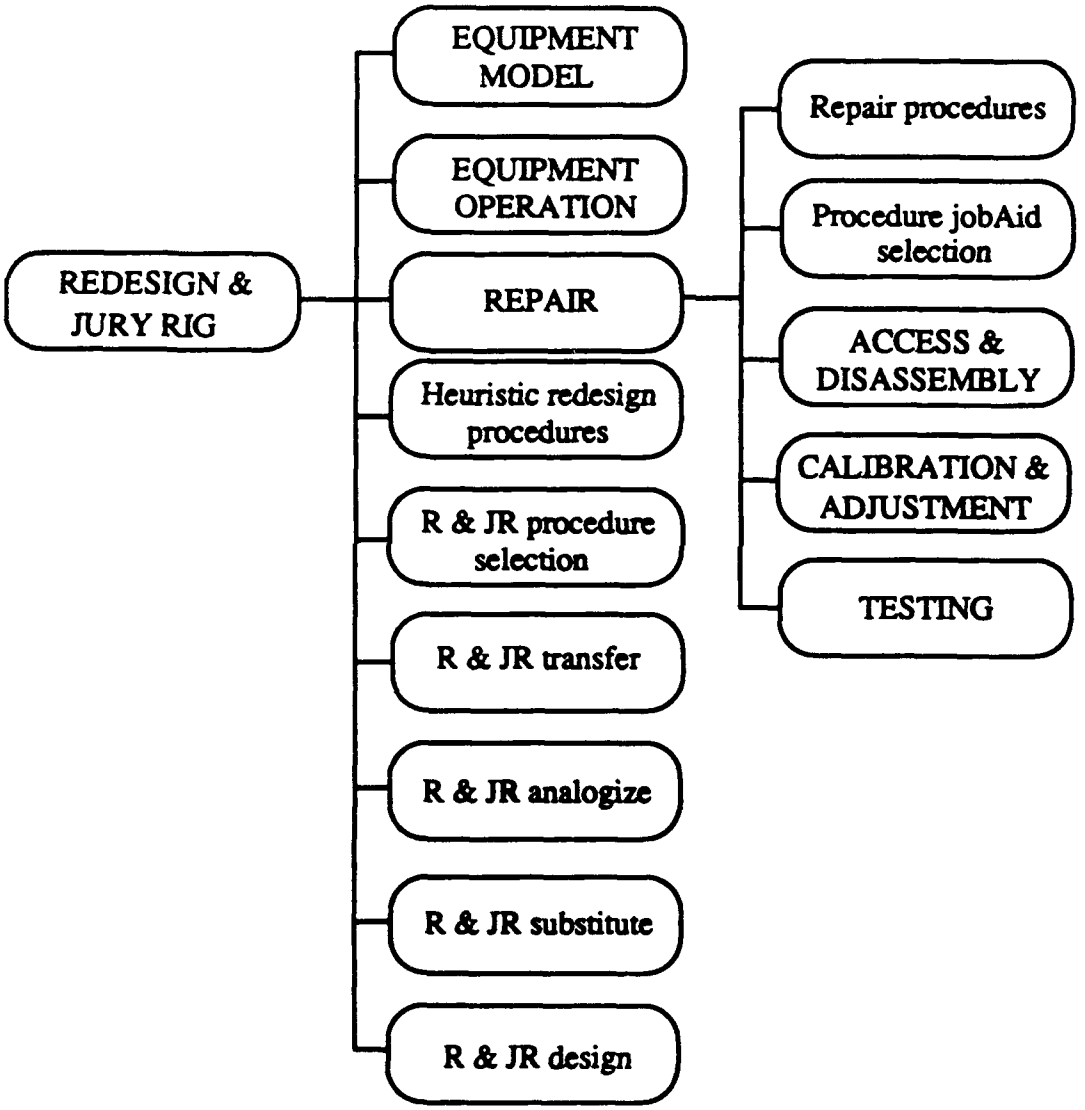


Figure 8. Transaction family for acquiring equipment redesign and jury rig enterprises

We have previously identified transaction classes. The specific transactions identified in each of the transaction families identified for maintenance training have been identified as to transaction class. This classification has considerable implications for the design of maintenance training transaction shells. First, all the transactions in a given class share considerable common features in terms of the nature of the interaction modes, their interaction strategies, etc. It should be possible to construct a common base transaction shell for each class and then customize this transaction shell to particularize this shell for the particular variations and transaction families of which it is a part. In addition, a particular transaction shell instance may occur in several families. Its interaction modes, interaction strategies, knowledge representation, and interaction parameters will vary depending on the family of which it is a part. Here again some development efficiencies can be realized by designing a common transaction shell and then developing variations of this shell for the various transaction families of which it is a part.

Maintenance training will use nine of the twelve classes of primary transactions: identify, interpret, execute, judge, decide/classify, transfer, analogize, substitute, and design. The following list identifies each of the transactions identified for the transaction families and categorizes them as to transaction class.

IDENTIFY:

(1) Physical and conceptual structure

INTERPRET:

(2) Device functioning

Performance

Device functioning enables the learner to explain the operation of a given device by knowing the sequence of events, and the conditions under which different events occur.

Knowledge required

This requires an operational controllable simulation of the device. The simulation should have both functional and some degree of structural fidelity.

Interactions

The transaction must present the operation of the device via a simulation that enables the learner to

change conditions and parameter values and observe the effects on the operation of the device. The transaction must be able to illustrate the operation of the device in a structured manner as well as enabling the learner to manipulate the operation.

(3) Device configuration

(4) Fault recognition

(5) Prediction

EXECUTE:

(6) Equipment operation procedures

(8) Calibration and adjustment procedures

(11) Testing procedures

(14) Access and disassembly procedures

(16) Repair procedures

(18) Logical fault isolation procedures

(19) Intuitive fault isolation procedures

(22) Heuristic jury rig procedures

JUDGE:

(9) Calibration and adjustment judging

(12) Testing judging

DECIDE CLASSIFY:

(7) Equipment operation procedure or jobAid selection

(10) Calibration and adjustment procedure or jobAid selection

(13) Test procedure or jobAid selection

(15) Access and disassembly procedure or jobAid selection

(17) Repair procedure or job aid selection

(20) Logical fault isolation procedure or jobAid selection

(21) Intuitive fault isolation procedure or jobAid selection

(23) Redesign or jury rig procedure selection

TRANSFER:

(24) Redesign or jury rig procedure transfer

ANALOGIZE:

(25) Redesign or jury rig analogies

SUBSTITUTE:

(26) Redesign or jury rig substitution

DESIGN:

(27) Redesign or jury rig redesign techniques

The following paragraphs identify the principal performance enabled by each of these specific primary transactions, a preliminary identification of the knowledge base required for each and a preliminary identification of the interaction modes required.

IDENTIFY:

(1) Physical and conceptual structure

Learning the names, location, and function of the parts of a device is a prerequisite to learning how a device works or how to operate a device.

Performance

"Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections" (Halff, p 17). Students are shown diagrams representing the conceptual structure of the equipment and asked to identify individual components, their function, and their immediate connections. Students are shown both the physical and conceptual representations and asked to demonstrate the correspondence between these two representations.

Knowledge required

The knowledge base required includes some representation of the device, probably a graphic

representation, with the parts isolated and an associated name and function available.

A graphical representation of the conceptual representation of the system involves pairing the conceptual symbols with the physical representation of the device.

Interactions

The transaction must both present the names to the learner and enable the learner to practice locating the parts and identifying the part name and function.

The transaction must present the conceptual names to the learner, must pair the conceptual names with their physical referents. The transaction must enable the learner to practice identifying conceptual symbols given referents; referents given the conceptual symbols; names of conceptual symbol given its graphic representation; and reproducing the symbol.

INTERPRET:

(2) Device functioning

Performance

"Students are asked to discriminate among component states on the basis of some physical depiction of those states" (Halff p. 17). The student can explain how the device functions, recognize the various states.

(3) Device configuration

Performance

"Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state" (Halff p. 17).

(5) Fault recognition

Performance

"Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted" (Halff p 17).

(5) Prediction

Performance

"Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode" (Halff p 17).

EXECUTE:

(6) Equipment operation procedures

Performance:

"Students are required to perform certain operational functions using both a physical and conceptual simulator. That is, each step in the procedure must be executed within the physical simulator and the conceptual simulator. For complex procedures the goal structure of the procedure should be tracked during procedure execution" (Halff, p 18).

(8) Calibration and adjustment procedures

Performance:

"Students work with a physical simulation of the device to practice required calibration and adjustment tasks. A conceptual simulation of the system being adjusted or calibrated shows relations among the components involved in the process" (Halff p 18).

(11) Testing procedures

Performance

"Students are required to carry out fixed testing procedures on a physical simulation of the equipment. A conceptual simulation of the components being tested is used to exhibit or query the student on the states of these components" (Halff p 18).

(14) Access and disassembly procedures

Performance

"Students are given the task of gaining access to a particular component. They use a physical simulation of the device to practice the task. A matching conceptual simulation shows which components are

accessible at each point in the procedure" (Halff p 18).

(16) Repair procedures

Performance

(18) Logical fault isolation procedures

Some forms of trouble shooting are best solved by acquiring an accurate operational model of the device or circuit that is malfunctioning and then systematically testing and or replacing components to eliminate potential trouble spots.

Performance

Logical fault isolation enables the learner to acquire an accurate functional model of the device or circuit. The learner acquires a set of systematic procedures for testing and/or replacing the components of the device or circuit.

"Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular faults or sets of faults" (Halff p 19).

Knowledge required

An accurate logical operational model of the device or circuit. A set of malfunctioning components which can be inserted into the device or circuit. A functional simulation of the device or circuit.

Interactions

The transaction must present the functional model of the device or circuit and enable the leaner to use this model in isolating faults that the system inserts into the functional model. The interactions must guide the learner through systematic fault isolation activities.

(19) Intuitive fault isolation procedures

Logical fault isolation is often less efficient than the use of a set of heuristic guidelines (rules of

thumb) to identify and correct faults in a device or circuit.

Performance

Intuitive fault isolation enables the learner to acquire a set of heuristic guidelines and to apply these guidelines in isolating a fault.

Knowledge required

An accurate logical functional model of the device or circuit. A set of malfunctioning components which can be inserted into the device or circuit. A functional simulation of the device or circuit. A set of heuristic guidelines for troubleshooting the device or circuit.

Interactions

The transaction must present the functional model of the device or circuit and enable the leaner to use this model in isolating faults that the system inserts into the functional model. The transaction must also present and enable the learner to acquire the heuristic fault isolation rules. The interactions must provide guidance which enables the learner to use the heuristics for fault isolation activities.

(22) Heuristic jury rig procedures

Performance

"Students are provided with conceptual simulations of tasks requiring complete or partial reconstruction of the equipment. For example, students could be required to restore as much functionality as possible with a limited inventory of spare parts or with other constraints on the reconstruction" (Halff p 18-19).

JUDGE:

(9) Calibration and adjustment judging

(12) Testing judging

DECIDE CLASSIFY:

Performance

The following performance applies to transactions 7, 10, 13, 15, 17, 20, 21, and 23. Each of these

transactions is a minor variation of the same transaction.

"Students are asked to identify the procedures needed to deal with particular situations and to select any appropriate job aids. Support is provided for this exercise in the form of subgoals and intermediate steps needed to arrive at the proper selection" (Halff p 18).

(7) Equipment operation procedure or jobAid selection

(10) Calibration and adjustment procedure or jobAid selection

(13) Test procedure or jobAid selection

(15) Access and disassembly procedure or jobAid selection

(17) Repair procedure or jobAid selection

(20) Logical fault isolation procedure or jobAid selection

(21) Intuitive fault isolation procedure or jobAid selection

(23) redesign or jury rig procedure selection

TRANSFER:

(24) Redesign or jury rig procedure transfer

ANALOGIZE:

(25) Redesign or jury rig analogies

SUBSTITUTE:

(26) Redesign or jury rig substitution

DESIGN:

(27) Redesign or jury rig redesign techniques

Integrating Simulations and Shells

While many shells will require simulations, these simulations need not be separately created. A single simulation may serve most or all shells for that content. For example, a simulation of the engine starting process can be used by an Interpret shell for teaching device operation, an Execute shell

for teaching the engine starting procedure, and an Identify shell for teaching nomenclature of the entities of the starting mechanism.

RAPIDS has instruction built into the simulation. The TRX approach would be to extract the instructional elements of RAPIDS out into shells, and then augment those shells with additional ones.

The following list shows the extraction and conversion for the RAPIDS instruction into shells.

Front panel orientation: Identify

Safety procedures: Execute

Theory of operation: Interpret

Fault diagnosis: Execute

Shell Parameters

Each shell has several parameters which configure the operation of the shell for a particular instructional instantiation. These parameters are set by the instructional designer. Examples of parameters for a naming transaction for the components of an entity follow.

Focus

Focus is a pointer to an entity frame in the EFN. The component hierarchy in which this frame participates will be instructed by the shell.

Content

Content indicates how much of the component hierarchy is available for instruction. The content parameter takes on one of the following values (default is All):

All:	entire hierarchy;
(list):	a list of frames (subset of the hierarchy) which are the content.

Coverage

Coverage indicates how much of the component hierarchy for the focus is to be instructed. The coverage parameter takes on one of the following values (default is Focus):

All:	entire hierarchy;
Focus:	the focus, its superpart (the single frame directly above the focus in the hierarchy), and its first level of subparts;
Levels:	requires an additional integer argument, which indicates how many levels of subparts of the focus are instructed;
Exemplar, <label>:	the focus and a single, specified subpart are instructed;
Random Exemplar:	the focus and a single, randomly selected subpart are instructed.

Guidance Level

This parameter sets the level of guidance provided to the learner and takes one of the following values (default is Full):

None:	no guidance;
OnDemand:	guidance presented on learner request only;
Full:	guidance at all times;
Faded:	begin with full guidance, fade to OnDemand by end of transaction.

Guidance Type

This parameter takes one of two values (default is Verbose):

Concise:	guidance interactions are short and to the point;
Verbose:	guidance interactions are detailed and complete.

View

The view parameter describes the representation of the subject matter. It takes one or more of the following values (default is Structural):

Structural:	displays the component relation for the knowledge structures, in tree format;
Physical:	displays an author-supplied graphic or illustration of the object whose parts are being instructed, representing the physical appearance of the object;
Functional:	displays an author-supplied graphic or illustration of the object whose parts are being instructed, representing the functional appearance of the object.

Vertical Sequence

The vertical sequence describes the order of introduction of the parts. The parameter takes one of the following values (default is TopDownBreadth).

TopDownBreadth:	ordering is from the highest superpart to the lowest level, breadth first (an entire level is introduced before any components of the next level are introduced).
TopDownDepth:	ordering is from the highest superpart to the lowest level, depth first.
BottomUp:	ordering is from the lowest subpart level to the highest, breadth first.

Temporal Sequence

Temporal sequence, within the vertical sequence, is the order of introduction of the parts on the same level. The default is LeftRight:

LeftRight:	ordering is accordingly to the representation in the EFN, from left to right;
LowHigh, <attribute label>:	ordering is according to the ranking of a named attribute, from low to high;
HighLow, <attribute label>:	ordering is according to the ranking of a named attribute, from high to low.

Trials

Trials defines the number of times to sequence through the set of parts. The parameter takes a positive integer value with a default value of 1.

Mastery Level

This parameter defines the percent correct required for mastery. It takes a value between 0 and 100. The default is 80%.

Response Mode

Response mode defines the type of response performance required of the learner. The parameter can take one of two values, either Recognition or Recall. The default is Recognition.

Feedback

The feedback parameter defines the timing of feedback. It takes one of the following values (default is PrePractice):

None:	no feedback is given;
PostResponse:	corrective feedback is given immediately after a response;
PrePractice:	corrective feedback is given just before the next opportunity to practice.

Replacement

Whenever sampling is used in the transaction, this parameter controls whether a new sample may or may not include items from previous samples. The default is With.

With:	a new sample may include items previously used;
Without:	new samples are distinct from previous samples.

Items

Whenever a pool of items is practiced or tested, this parameter sets the maximum size of the pool. It takes a positive integer value. The default is 3.

Timeout

This parameter is the amount of time to wait for a user response before timing out. If the user response involves typing rather than pointing, the timeout occurs after a base interval, plus a fraction of the base interval multiplied by a number derived from the length of the expected response. If the user response is pointing only, the timeout occurs after the base interval. The value of the parameter is the base interval, a positive integer. The default is 3 (seconds).

Item Order

Whenever a pool of items are practiced or tested more than once, this parameters controls whether the ordering is the same or different. The default is Random.

Random:	the ordering is random;
Same:	the ordering is fixed.

Modes

The mode is the method of interaction with the student. One or more of the following may be selected (default is Overview):

Overview:	presents the knowledge structure from the knowledge base in the Structural view;
Presentation:	presents an author-supplied graphic in either the Physical or the Functional view, and demonstrates the parts to the learner;
Practice:	provides practice for the learner using the author-supplied graphic, in either the Physical or Functional view;
InstanceAssessment:	tests the student's mastery of the material, using the author-supplied graphic, in either the Physical or Functional view.

Strategy

Strategy is defined as a sequence of modes. Because modes are fully determined by strategy, the arguments to the Mode parameter are ignored unless the Strategy parameter is None (the default):

Overview:	the Overview mode;
Familiarity:	the Overview mode followed by the Presentation mode;
Basic:	Overview plus Presentation plus Practice modes;
Mastery:	Overview plus Presentation plus Practice plus InstanceAssessment;
BasicRemediation:	Instance Assessment, followed by Basic Strategy to remediate errors;
MasteryRemediation:	InstanceAssessment, followed by Mastery Strategy to remediate errors;
Assessment:	Instance Assessment mode;
Summary:	Overview with Coverage set to Focus, plus Presentation followed by Instance Assessment, with Coverage set to Random Exemplar for both;
None:	no strategy.

Strategic Control

Strategic control determines the level of control granted the learner over the selection of strategy, mode, and content. It takes one of the following values (System is the default):

System:	the strategy, mode, content, and coverage are delivered as set by the parameter values;
Learner:	the learner may select alternate strategies, mode, content, and coverage.

Tactical Control

This parameter determines control over the initiation and termination of interactions. It also determines whether the learner may alter the values of the Guidance, View, and Sequence parameters. Tactical control takes one of two values, System or Learner. The default is System.

Configuring Shells

Configuring is the setting of parameters to a shell, and attaching content. For example, a call to a shell in the Identify class to instruct the names and locations of the left engine starting circuitry might be configured as follows:

focus:	left engine starting circuitry
content:	all
coverage:	all
guidance level:	faded
guidance type:	concise
view:	structural, functional
vertical sequence:	topDownBreadth
temporal sequence:	leftRight
trials:	1
mastery level:	100%
response mode:	recall
feedback:	postResponse
replacement:	without
items:	3

timeout:	3
item order:	random
strategy:	basic
strategic control:	system
tactical control:	learner.

Detailing Shells

Detailing is the attachment of graphics and text, prepared offline, to the knowledge base for use by the shells. With the use of simulations, detailing requirements are replaced largely by simulation authoring.

Strategy Analysis

The Strategy Analysis step identifies the enterprises to be learned, and selects and sequences transactions to instruct the enterprises. Additionally, information about the audience and the instructional setting are gathered in this phase.

Enterprise Transactions

An enterprise is a complex human performance that requires an integrated set of knowledge and skills. The goal of instruction is the acquisition by the learner of one or more enterprises.

The primary transaction shells, previously described, facilitate the acquisition of the knowledge and skills which comprise enterprises, but by themselves cannot accomplish their integration. This integration must be accomplished by transactions at the enterprise level.

An enterprise transaction accomplishes two principal purposes. First, it functions as a transaction manager, providing the overall direction of the execution of the primary transaction instances which instruct the knowledge and skills necessary to the enterprise. Second, it provides for an integration of the learning facilitated by the primary transactions, ideally in the context of a performance or simulation of an authentic activity that is representative of the real-world performance of the enterprise.

The integration is accomplished by focusing the enterprise on a particular performance that is integrative of the elements of the enterprise. The primary transaction that directly instructs the integrative performance becomes the focus

transaction for the enterprise. Other transactions are introduced to support the performance on the focus transaction.

A course, then, is defined to be a set of enterprise transactions, and their supporting families of primary transactions. A course organization is a nesting and/or ordering of the enterprise transactions.

Classes of Enterprise Transactions

Different types of enterprises can be discriminated on the basis of the level of performance required and the type of knowledge involved with this performance. We have identified six classes of enterprises: *denote, evaluate, execute, design, interpret,* and *discover*. This class structure may also be used to classify the enterprise transactions according to the class of enterprise being facilitated.

A Denote enterprise transaction requires as a focus one of the following: a primary transaction from the Component class, either an Identify transaction for an entity, an Execute transaction at the Denote level of performance for an activity, or an Interpret transaction at the Denote level of performance for a process frame; or a Classify/Decide primary transaction from the Abstraction class. (Performance level is a parameter to Execute and Interpret transaction shells. For Execute, the values may be either Denote or Perform, for Interpret, values are either Denote, Explain, or Predict.) Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for a denote enterprise is characterized as *knowing about* something. With a Component class primary transaction as focus, the enterprise transaction enables the student to describe the parts, their functions and locations for an entity; describe the steps for an activity, or describe the events for a process. With a Classify/Decide primary transaction as focus it enables the student to identify instances or discriminate kinds.

An Evaluate enterprise transaction requires, as a focus, a Judge primary transaction, of the Abstraction class. The Judge transaction instructs an abstraction hierarchy of either entities, activities, or processes. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for an Evaluate enterprise is characterized as *classifying and ranking* the adequacy of an entity, the performance of an activity, or the effectiveness of a process.

An Execute enterprise transaction requires, as a focus, an Execute primary transaction (at the Perform level) from the Component class. The content for the focus transaction is the steps of an activity frame. Primary transactions from the

component, abstraction, and association classes would be included to support the focus transaction. Performance for an Execute enterprise is characterized as *performing* some activity.

A Design enterprise transaction requires, as a focus, a Design primary transaction from the Association class. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for a Design enterprise is characterized as *inventing or creating* a new artifact. It enables the student to design a new entity or activity not previously instructed.

An Interpret enterprise transaction requires, as a focus, an Interpret primary transaction, at the Explain or Predict level of performance, from the Component class. The content for the focus transaction is the events and causal network of a process frame. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for an Interpret enterprise is characterized as *knowing why* some process works.

A Discover enterprise transaction requires, as a focus, a Discover primary transaction from the Association class. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for a Discover enterprise is characterized as *finding* a new relationship or process. It enables the student to discover a new entity or process not previously instructed.

Authoring Enterprise Transactions

The enterprise transaction is responsible for integrating the instruction of the knowledge and skills necessary to the enterprise. This integration is accomplished by the selection of a focus transaction that instructs an integrative performance, and by sequencing the focus transaction in conjunction with the supporting primary transactions.

Course organization comprises the sequencing of the enterprise transactions themselves, plus the sequencing of primary transactions within each enterprise transaction.

Sequencing Alternatives

There are two dimensions of sequencing at the enterprise level, yielding seven sequencing alternatives. The first dimension, Primary Sequence, includes Encyclopedic, Case Study, and Situational.

The *encyclopedic* sequence systematically calls each primary transaction to instruct elements of the content, eventually

integrating these at the enterprise level. This type of sequencing is often found in textbooks and reference manuals.

The *case study* sequence presents a sequence of carefully selected examples, scenarios, or cases of the focus transaction and the necessary supporting transactions, with each case being complete in and of itself. The sequence of cases is graded on some dimension, such as familiarity, frequency, or criticality.

The *situational* sequence is characterized as on-the-job learning, where instruction is delivered on an as-needed basis. Only that instruction necessary to the immediate task is presented; integration must occur opportunistically. Situational sequence is facilitated by an online advisor system and student modelling.

The second dimension, which we call Secondary sequence, includes Elaboration, Prerequisite, and Flat sequence.

Elaboration sequence starts with a simple, representative element or elements of the focus content, and progressively adds layers of detail as the instruction progresses. This is similar in many respects to Riegeluth's Elaboration Theory.

Prerequisite sequence orders elements of subject matter based on their dependency interrelations. This is based on Gagné's learning hierarchies. The focus content is at the top level of the hierarchy.

Flat sequence involves no systematic ordering at the secondary level.

The primary and secondary sequences may be combined into seven approaches to sequencing: elaborated, prerequisite, and flat case study; elaborated, prerequisite, and flat encyclopedic; and situational.

Elaborated case study requires a number of cases of the focus transaction, each complete in of it itself. In our earlier example of a Circuit Functioning enterprise transaction, the focus transaction was an Interpret primary transaction to instruct circuit functioning. Suppose that the specific enterprise involved the functioning of AC circuits. The enterprise would require a set of cases, each of which would be instructed by the focus transaction. The cases would be drawn from an abstraction hierarchy of circuits, and would be ordered on some relevant dimension, such as complexity, familiarity, frequency of occurrence, etc. Examples might include specific instances of capacitance reactive circuits, resonant circuits, and transformers. Each case would be an instance of a class in the abstraction hierarchy. However, instructing the abstraction hierarchy is not the focus; rather it is the interpretation of

circuit functioning which is the focus. The instructing of the abstraction hierarchy is supporting instruction to the focus.

As each case is selected in turn, it is introduced by the focus transaction to the student, following an elaboration secondary sequence. Other information would then be brought into the instruction, from the focus and from supporting transactions, until the circuit had been fully instructed. The next case would then be presented, refreshing and reviewing content that had already been introduced in earlier cases, and introducing additional content.

Prerequisite case study selects cases equivalently, but the secondary sequence follows a prerequisite hierarchy. The case would be overviewed by the focus transaction, but then instruction would build bottom-up following the prerequisites. Each supporting transaction might be called one or more times at different nodes in the hierarchy. Then the next case would be handled in a similar way, refreshing and reviewing content that had already been introduced in earlier cases, and introducing additional content.

Flat case study has no systematic secondary ordering. Once a case had been selected, instruction begins with an overview from the focus, then each supporting transaction would be called in turn to present all required content for that case, finally returning to the focus for a full presentation. Then the next case would be selected.

The encyclopedic sequences are not built on cases. Any abstraction hierarchy is taught as part of the supporting content, rather than being used to generate cases.

Elaborated encyclopedic sequence begins with a representative element or elements of the focus content, introduces supporting content as needed, then builds to the full focus content.

Prerequisite encyclopedic sequence begins with an overview of the focus content, then goes to the lowest levels of a prerequisite hierarchy and sequences the primary transactions to deliver instruction for nodes on the hierarchy, building eventually to full focus transaction.

Flat encyclopedic sequence begins with an overview of the focus, then each supporting transaction would be called in turn to present all required supporting content, finally returning to the focus for a full presentation.

Situational sequencing delivers instructional elements on demand, either as a result of user request or based on an online

determination by an advisor program of the learning requirements of the user.

Making Sequence Decisions

Authoring the sequence for an enterprise transaction involves the following steps:

1. determine enterprise content;

2. select the focus transaction;

3. for each content grouping, select supporting transactions;

4. select primary and secondary sequence;

5. if case study, create instances for the focus content;

6. identify content for each case;

7. if prerequisite sequence, identify prerequisite relations;

8. if elaboration sequence, identify elaboration levels; and

9. configure parameters for each call to a transaction.

We now examine the roles of the author and the system for performing each step.

Step 1, determine enterprise content, begins by the author selecting the focus content, such as an activity or process frame. The system then initiates a spreading activation search of the EFN, following relations from that frame. For each relation leading from the frame, the author indicates whether that relation should be included in the enterprise. The search continues from the related frames for any relation that is included; while a path is terminated for any relation not included. This continues until all paths have either reached their end or been terminated. The system then creates the enterprise transaction structure, and stores a representation of the subset of the EFN that has been selected as the content for the enterprise.

Step 2, select the focus transaction, is performed by the author based on a recommendation by the system. The recommendation is based upon the transaction class appropriate for the content structure of the focus.

Step 3, select supporting transactions for each content grouping, is performed by the author with consultation from the system. The system parses the content into content groupings according to the classes of primary transactions (component and abstraction hierarchies, and association links). Each grouping is presented in turn to the author, along with recommended transactions for that grouping. Recommendations are based first on the transaction class appropriate for the content grouping, and may be further refined by environmental parameters, such as the availability of specific resources. Recommendations may also take into account compatibility with the focus transaction. For example, if the selected focus transaction uses video, employs a particular instructional technique, or is optimized for a given domain area, then the recommended supporting transactions will take this into account. Recommendations are ranked if there is more than one possible choice.

Step 4, select primary and secondary sequence alternatives for the enterprise transaction, is performed by the author based on recommendation of the system.

Step 5, if case study, create instances for the focus content. The author selects an abstraction hierarchy from which cases will be drawn, and identifies the attribute which will be used to order cases. The system then prompts the author to return to knowledge analysis to identify the instances for the classes in the hierarchy which will form the cases.

Step 6, identify content for each case, is performed automatically by the system by parsing the subset of the EFN selected in step 1, selecting any content related to the focus or the instance.

Step 7, if prerequisite sequence, identify prerequisite relations, is performed by the author using a system tool to identify dependency relations. This relation structure is stored with the enterprise. If prerequisite case study, the prerequisite relations for each case may be derived automatically from this structure.

Step 8, if elaboration sequence, identify elaboration levels, is performed by the author. The number of levels, and the content for each level of elaboration, is identified. If case study, this is performed for each case. This data is stored with the enterprise transaction. Secondary content for each level of elaboration will be sequenced by the prerequisite relations, if available, or flat. At this point, all primary and secondary sequencing has been completed.

Step 9, configure parameters for each call to each transaction, is performed by the author. The system brings up the configuration for each call in turn, and presets as many

parameters as possible. These include the content for the call, based on the earlier steps, and the values of other parameters based on either student attributes and/or earlier configuration decisions for that enterprise. In addition to the normal configuration capabilities, the author may set a parameter for all calls to the transaction from this enterprise, for all calls to any transaction having the parameter from this enterprise, of for all calls to any transaction having the parameter from this course. (An example of the latter might be setting display or response parameters to establish a uniform interface across the course.)

Example Strategy Analysis for T-38 Maintenance Training

The example is for the engine starting procedure.

Step 1. Determine enterprise content. The focus content is the abstract activity Start_engine. The remainder of the content would be selected by following links in the knowledge base from that frame.

Step 2. Select the focus transaction. The appropriate transaction is from the primary class Execute.

Step 3. Select supporting transactions. At a minimum, transactions from the classes Identify (for the devices) and Interpret (for the processes) will be required.

Step 4. Select primary and secondary sequence. Case study is selected as the primary sequence, elaboration as the secondary sequence.

Step 5. Create instances for each case. At minimum, three cases are identified: ground start, emergency start, and in air start.

Step 6. Select content for each case. The author performs this with guidance from the system, by traversing the EFN. For example, the content for an Identify transaction for the instrument panel would require the names and locations of all instruments used in starting procedures.

Step 7. Identify prerequisite relations. Even though the secondary sequence is elaboration, prerequisite relations are needed for ordering supporting content. The author identifies these relationships among the supporting transactions. For example, knowing the locations and names of the controls would be prerequisite to performing the engine start activity.

Step 8. Identify elaboration levels. For each case, one or more elaboration levels may be identified. For example the

ground start activity may first be taught with no glitches, then with one or more problems introduced.

Step 9. Configure parameters to each call to a transaction shell. This is performed by the author using the transaction configuration system.

Instructional Delivery

This step describes the actual delivery of instruction to the student.

Instructional Modes and Strategies

The type of transaction and the components of its knowledge base limit the interactions that are possible within a given transaction shell. Different classes of transactions will have different types of interactions. Nevertheless, all transactions should include interactions that are characterized by certain interaction modes. Interaction mode alternatives determine the method of interaction with the student. Interaction modes assume different values on the form of the interaction (expository or inquisitory) and the degree of learner control involved (learner control or system control). Five interaction modes have been identified: *overview, presentation, practice, generality assessment,* and *instance assessment*.

Overview mode presents the knowledge structure, as represented in the EFN. For example, in an Identify transaction, overview would show the parts hierarchy of an entity in tree format. Text instruction may accompany the diagrams. The overview serves as an advance organizer, and as a review.

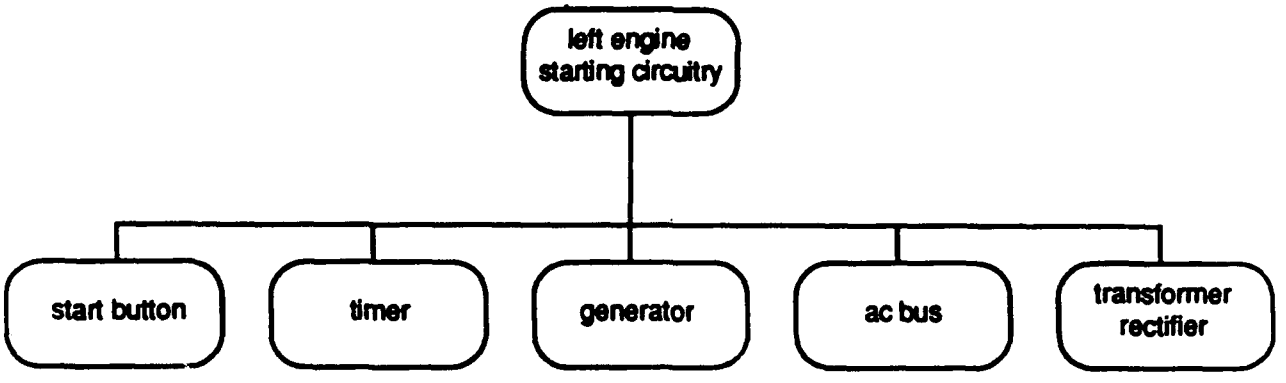


Figure 9. Example Overview mode for an entity component hierarchy

Presentation demonstrates and presents the content represented by the knowledge structure, *in terms of* both generalities and instances. For example, an Interpret transaction for device operation would simulate the operation of the device, explaining the events associated with the process.

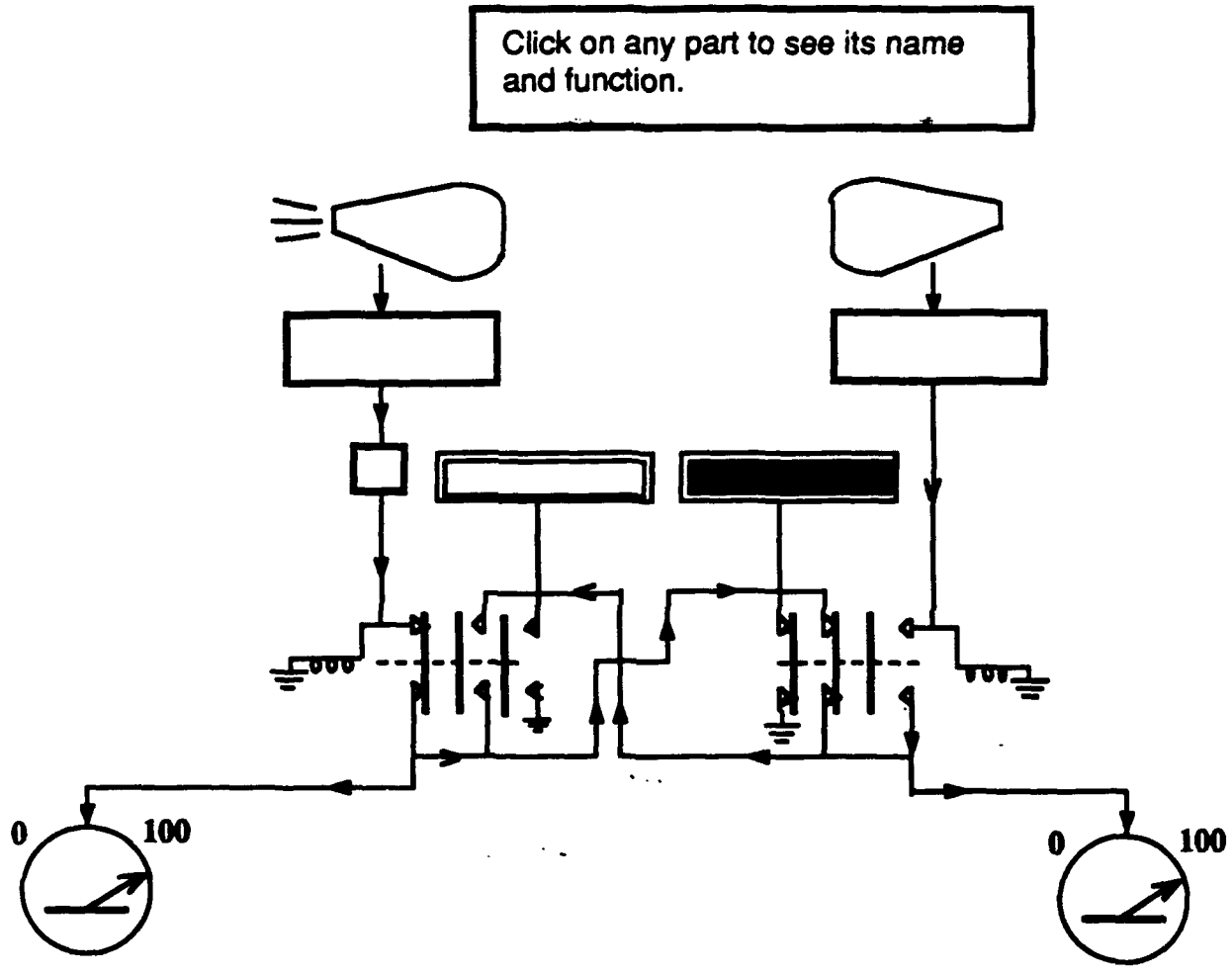


Figure 10. Example Presentation mode for an Identify transaction

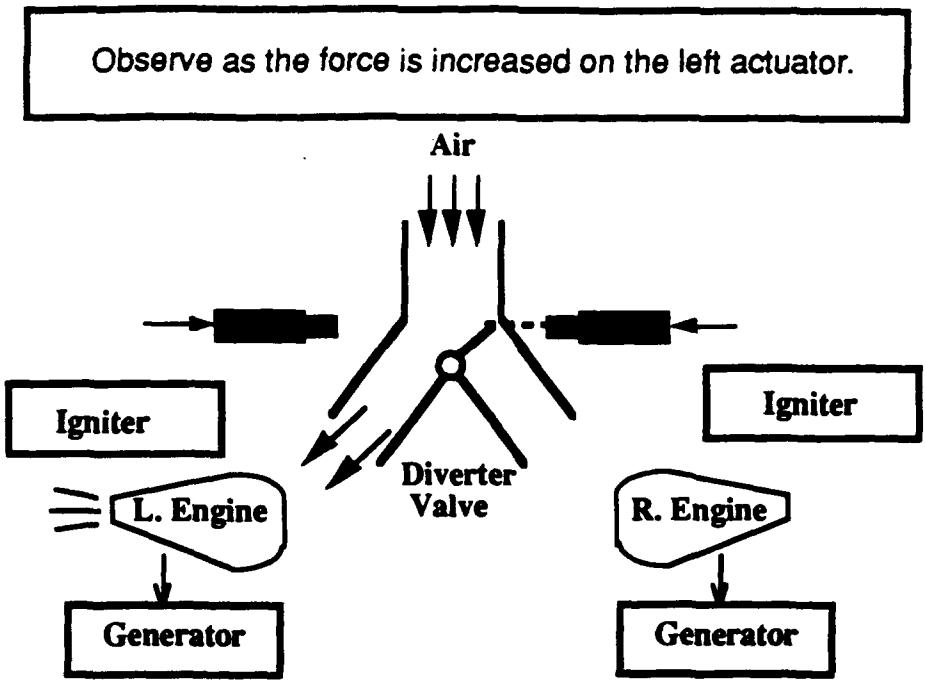


Figure 11. Example Presentation Mode for an Interpret transaction

Practice provides opportunity for the learner to work with the content directly. For example, an Execute transaction for an activity would provide a simulation which could be manipulated by the learner, with the consequences of actions displayed. Practice for an Interpret transaction would allow the learner to adjust controls, regulate inputs, and modify the functioning of devices, and to predict the consequences of these actions.

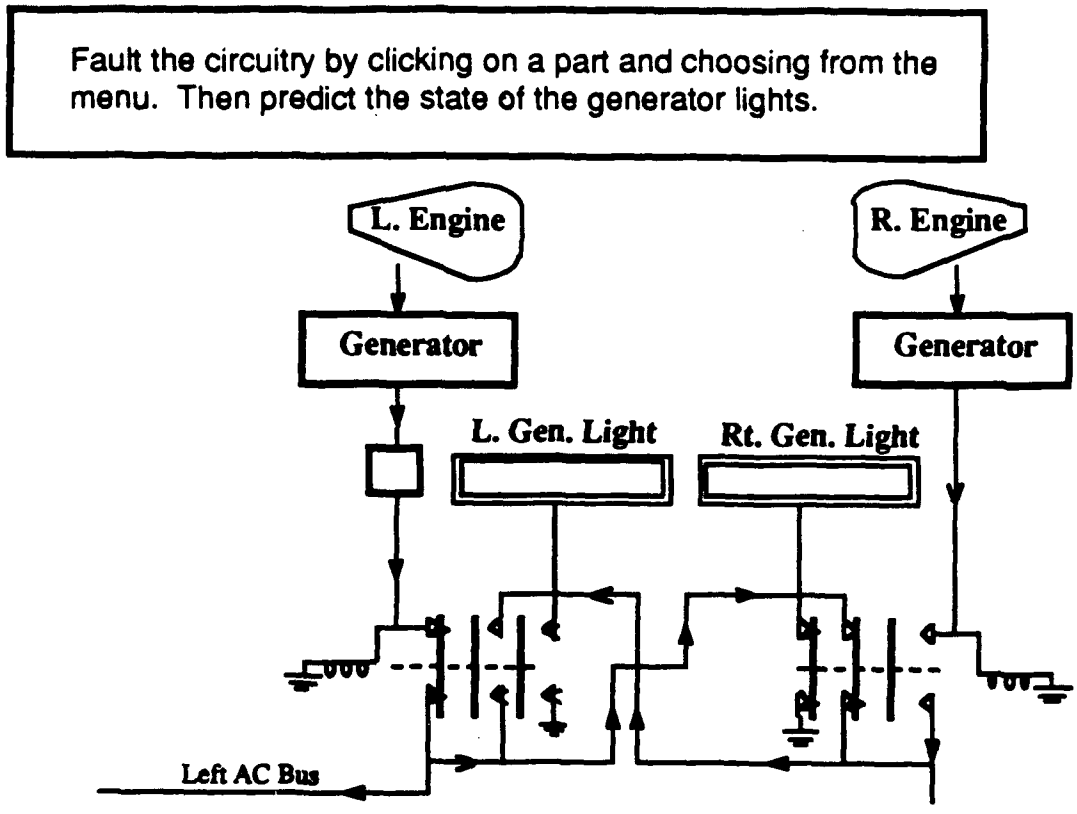


Figure 12. Practice Mode for an Interpret Shell

Generality and instance assessment test at the generality and instance level, respectively. Test results are recorded by the delivery system, under parametric control of the transaction shell.

Interaction strategy is the combination and sequence of interaction modes available to the student. We have identified seven interaction strategy alternatives: *overview, familiarity, basic, mastery, basic remediation, mastery remediation, summary,* and *assessment*.

Overview consists of the overview interaction mode.

Familiarity consists of an overview interaction plus a presentation.

Basic instruction consists of an overview plus presentation plus practice.

Mastery instruction consists of overview plus presentation plus practice plus generality and/or instance assessment; if the criterion is not met a new presentation, practice, and assessment for missed items is engaged until the criterion is met.

Basic remediation consists of generality or instance assessment; if the criterion is not met than basic instruction is provided for the missed items.

Mastery remediation consists of generality or instance assessment; if the criterion is not met then mastery instruction is provided for the missed items until the criterion is met.

Summary is an overview plus presentation followed by instance assessment; both the presentation and the assessment are for a single representative element of the knowledge structure, rather than the full knowledge structure.

Assessment consists of generality or instance assessment.

Interaction of Simulations and Shells

In order to integrate the shells with the simulations at delivery time, there must be a communications interface established. This interface would establish conventions whereby shells would be able to:

query simulations to determine their capabilities;

query status information from simulations;

issue commands to simulations to set the simulation directly into a state; and

replace direct learner input with commands from the shell.

Online Delivery Advisor

The authoring decisions made at design time are based on the designer's best estimate of the learner population. During the delivery of instruction, information about the learner, his or her aptitude, specific goals, motivation, familiarity, and other factors, as well as the learner's expressed preferences, may be taken into account to modify those decisions.

An online delivery advisor would have access to the domain knowledge base and the configurations. In addition, it would maintain a student model that contained information about the learner. Using the information gathered about the student, the

advisor would adjust design decisions to customize the instruction to more adequately meet the characteristics of the student. The advisor could also engage in a mixed-initiative dialog with the student which would allow the student to participate in this decision-making.

Example Instruction for T-38 Maintenance Training

The instructional capabilities of RAPIDS, as mentioned, would be captured into transaction shells of the classes Identify, Execute, and Interpret. Instruction would be extended in those shells in the following ways:

instruction that presented the structure of the knowledge would be incorporated, for example, displaying the parts of a device in tree format;

the parameters governing practice and test would be included, for example, item order, replacement and feedback;

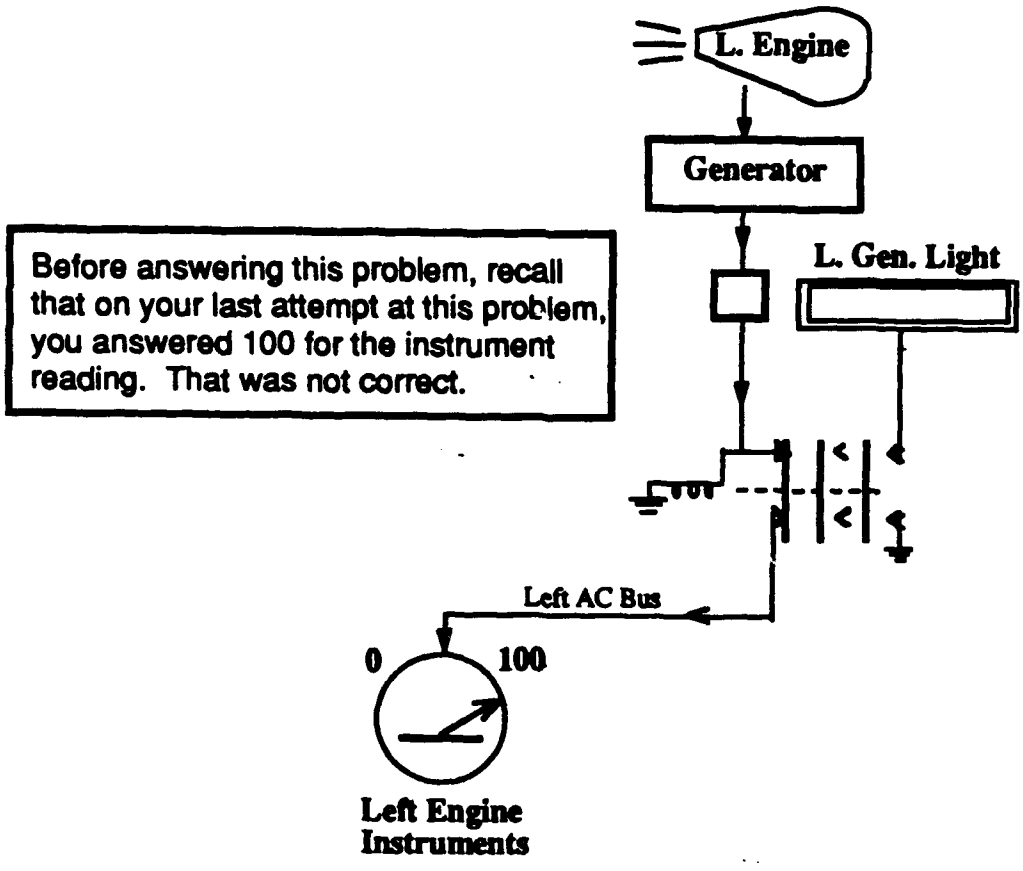


Figure 13. Example of pre-practice feedback

the capability to invoke a transaction for a specific purpose would be included, such as a single example, or a summary;

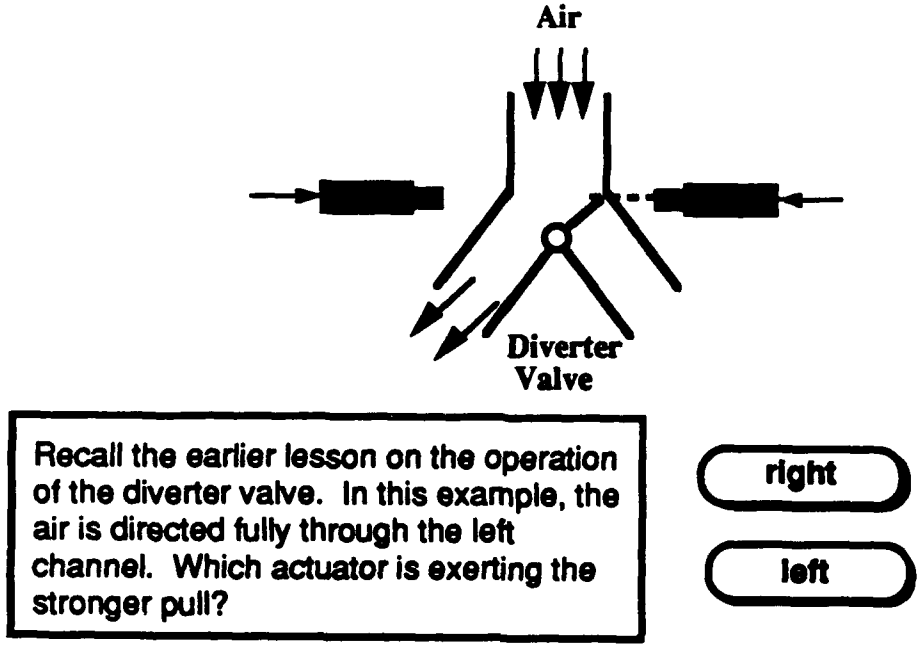


Figure 14. Example of a shell being called under Summary Strategy. A single problem is presented to test whether the student needs a refresher

the instruction would also include system-control presentation, practice, and assessment; and

integration with other transactions via the enterprise transactions, and the structuring of the instruction according to primary and secondary sequencing.

Additional shells would be provided for the classes of Judge, Decide/Classify, Transfer, Analogize, Substitute, and Design. Sample displays from these transactions are shown below.

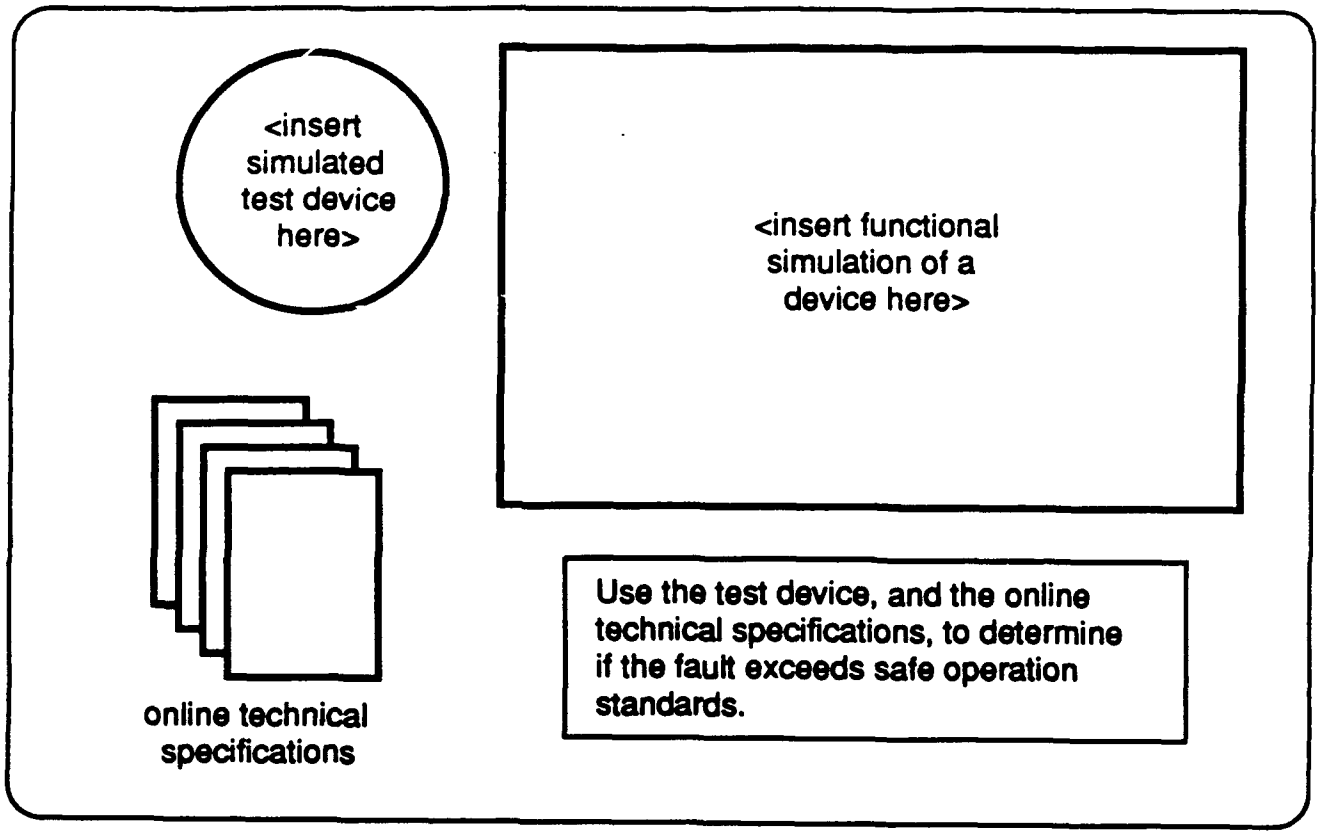


Figure 15. Example interaction from a Judge transaction

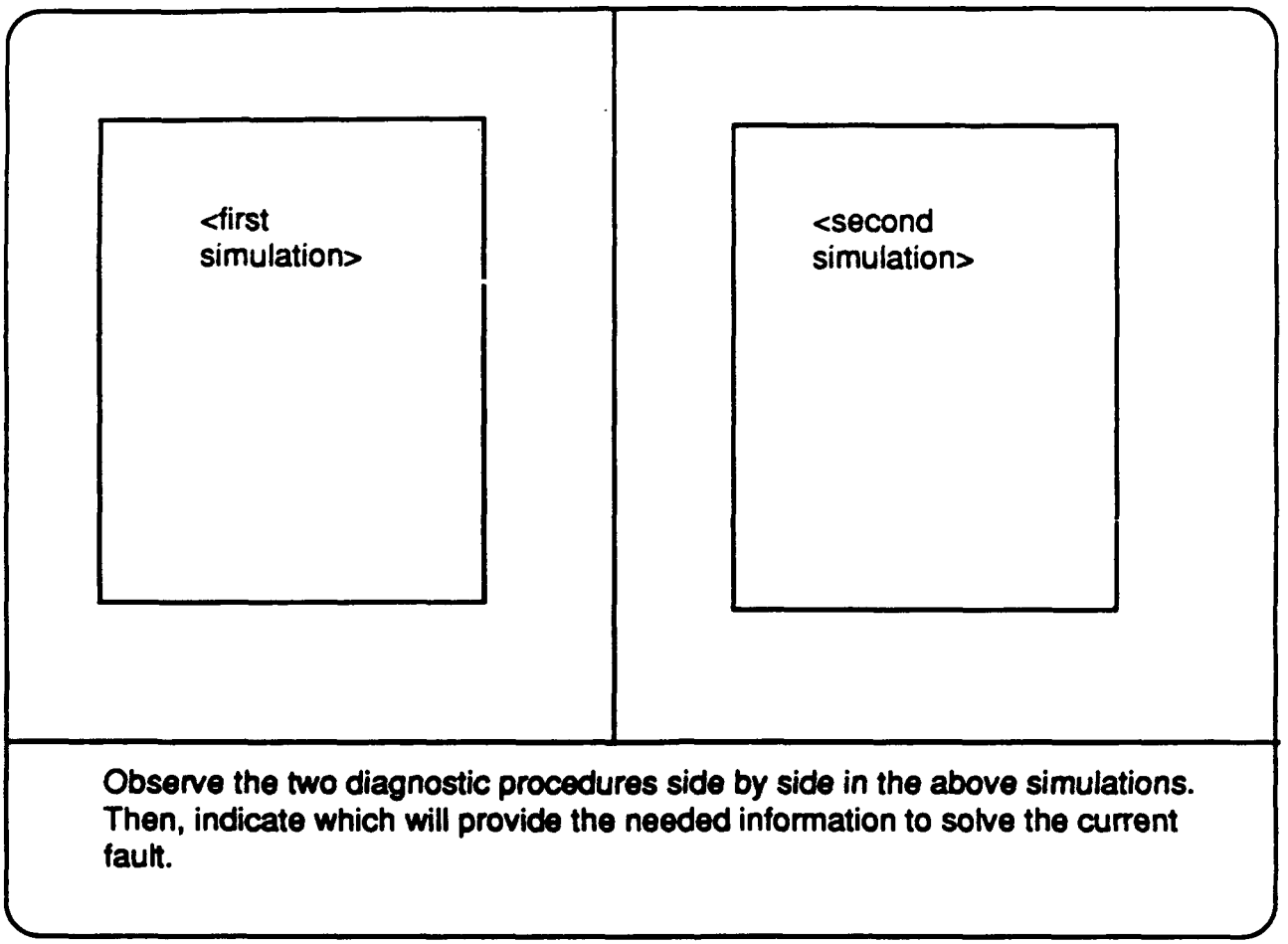


Figure 16. An interaction from a Classify/Decide transaction for comparing and contrasting related activities

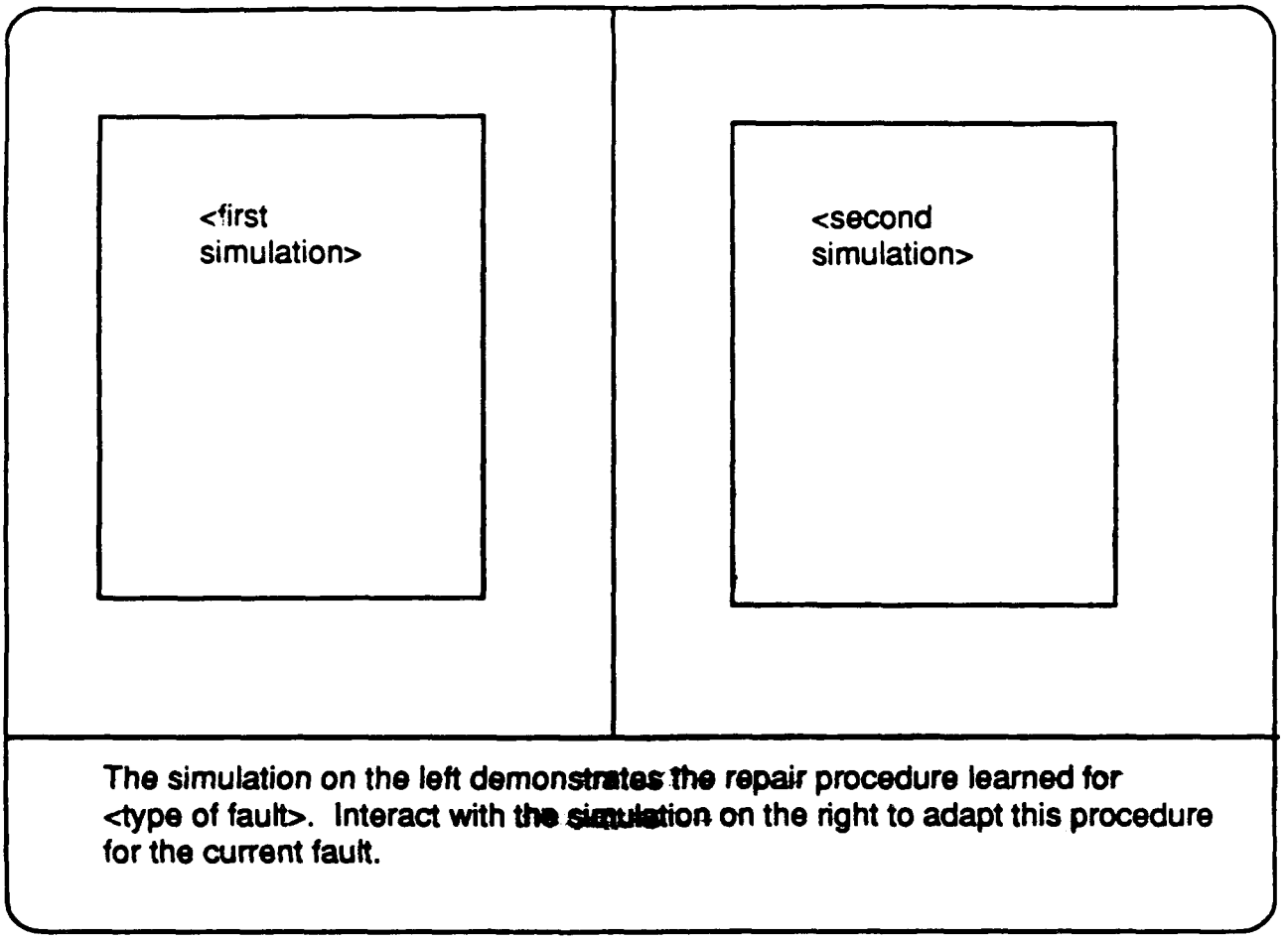


Figure 17. Example interaction from a Transfer transaction for repair procedures

III. RAPIDS AND AIDA (TOWNE)

RAPIDS was developed by Behavioral Technology Laboratories, University of Southern California, under funding from the United States Air Force Armstrong Laboratory, Intelligent Systems branch. Dr. Wesley Regian served as scientific officer. Support for early development of RAPIDS (then called IMTS) was provided by the Office of Naval Research and the Navy Personnel Research and Development Center.

The walkthrough illustrates the steps required to produce instruction using the RAPIDS system, and the instructional interactions that result. The instruction shown being developed is intended for enlisted personnel who maintain the T-38 dual-engine jet aircraft, focussing primarily on problems in the engine-starting functions. The instruction includes portions of the safety procedures required by all personnel working on the aircraft.

The walkthrough is a close depiction of RAPIDS as it currently exists, although **some of the design changes planned for the next generation system (386-based, RIDES system) are reflected.**

The walkthrough is presented in four sections:

1) specifying the course structure;
2) building the device simulation;
3) producing the instruction; and
4) delivering the instruction.

Sections 1 and 2 are written from the point of view of an instructional planner (IP), section 3 illustrates the activities of the subject matter expert (SME), and section 4 shows a sampling of the instructional events seen by a learner. Time estimates are given for each step of the process, to provide some idea of the level of effort required at each stage. These are rough estimates, based upon prior experience, and are intended to be realistic.

Specifying the Course Structure

<u>Produce the Course Plan (8 hours IP, 8 hours SME)</u>

The IP and SME, decide what the instructional objectives are, what the components of instruction will be, the order in which they will be presented, the proficiency criteria required for each, and the general instructional strategy. The topics and the maximum times allowed the learner for each unit of instruction are as follows.

Front panel orientation
- Locations & functions (1 hr. Instruct, 1 hr. Drill, 1/2 hr. Test)
- Naming Drill (1/2 hr. Instruct, 1/2 hr. Drill, 1/4 hr. Test)

Safety procedures (1 hr. Instruct, 1/2 hr. Drill, 1/2 hr. Test)

Theory of operation
- Generation of AC (1/2 hr. Instruct, 1/4 hr. Drill, 1/4 hr. Test)
- Cross-over system (1/2 hr. Instruct, 1/2 hr. Drill, 1/4 hr. Test)
- In-air start (1 hr. Instruct, 1 hr. Drill, 1/2 hr. Test)
- Emergency start (1/4 hr. Instruct, 1/4 hr. Drill)
- On-ground start (1 hr. Instruct, 1/2 hr. Drill, 1/4 hr. Test)

Fault Diagnosis
- Testing
  - Fault Effect Drill (given a failure, what would be abnormal)(4 hr. Instruct, 4 hr. Drill)
  - Test Evaluation Drill (Normal/abnormal)(1 hr. Instruct, 1 hr. Drill)
  - Symptom Interpretation Drill (possible causes of symptoms)(4 hr. Instruct, 4 hr. Drill)
  - Test Selection Drill (given possible failures, pick a test)(2 hr. Instruct, 1 hr. Drill)
- Failure effect free exploration (2 hours)
- Practice problems (20 failures x 1/233 hour)

Review of safety procedures (1/2 hr. Drill, 1/2 hr. Test)

The plan is to start out with front panel orientation (locations and names of controls and indicators), followed by some safety procedures. Then, theory of operation of the system is instructed, followed by training in fault diagnosis. The course ends with a review of safety procedures.

The diagnosis training includes part-task exercises in recognizing and interpreting effects of faults, evaluating front panel indications (making normal/abnormal assessments), interpreting symptom information (identifying implicated and absolved functions), and finally making effective test selection decisions. Then, the student will be allowed to explore normal and abnormal conditions of his or her own choosing, to discover and resolve areas of confusion. The final unit presents practice troubleshooting problems, supported with assistance and guidance as required for each student.

The SME uses the graphical course planning tools to enter the course structure, as shown in Figure 18.

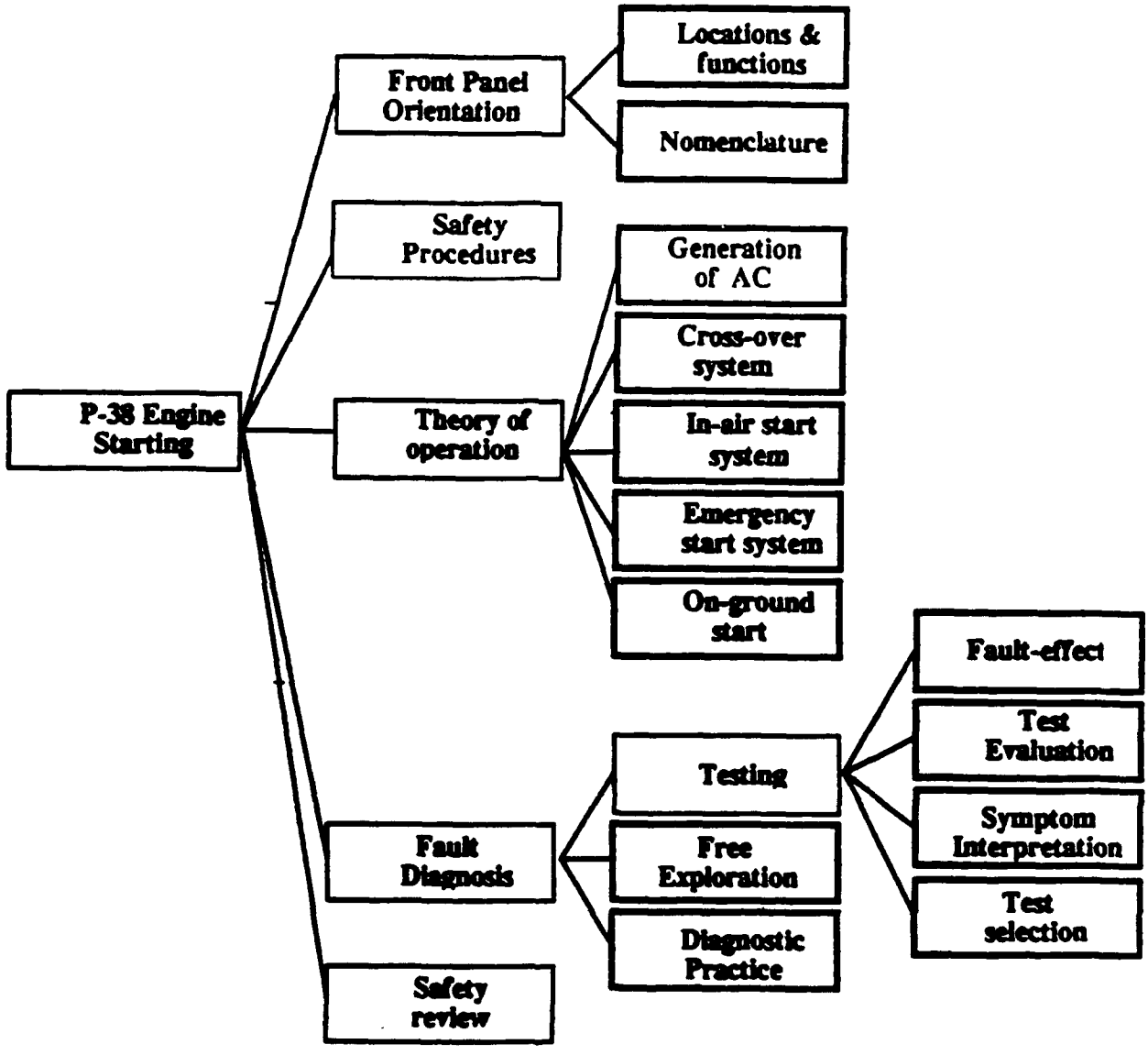


Figure 18. The RAPIDS Course Plan

Then, the SME enters the parameters that establish the minimum and maximum time to allocate to each instructional unit and the proficiency required of the learner to proceed. The parameters entered for the Location/Function unit are shown in Figure 19.

Unit Name: **Location/Functions**

Weight	Mode	Condition	Maximum Tries	Minimum Tries	Maximum Time	Accuracy Criterion	Speed Criterion
1	INSTRUCT	-	3	1	60	-	-
1	DRILL	-	3	1	60	0.95	10
1	TEST	-	1	1	30	0.95	10

Figure 19. Parameters for Controlling Instruction

Building the Device Simulation

Study the technical materials, and plan the physical and functional simulation (8 hours IP, 8 hours SME)

Physical Equipment Model

The T-38 Flight Manual (T.O. 1T-38A-1) provides static figures that illustrate the organization of the system and form the basis for simulating the physical system. The following figures, in particular, represent the sections of the aircraft pertinent to instruction on engine ignition.

Figure 1-1 General Aircraft Arrangement
Figure 1-4 Throttle Quadrant
Figure 1-5 Cockpit Arrangement - Front
Figure 1-7 Instrument Panel
Figure 1-9 Left and Right Subpanels
Figure 1-14 Caution Light Panel

The first two of these six figures involve considerable artwork, which has already been done professionally in the flight manual. Thus, these figures are scanned in using a digital page-scanner, and will appear as shown in Figures 20 and 21. Computer-graphics will be added to the throttle figure, to allow the student to operate it.

The remaining figures will be constructed of graphic objects using the RAPIDS simulation-construction tools. For this instruction, only two indicators on the Instrument Panel need to be provided in detail, the two engine tachometers. If other instruction will be conducted, the instrument panel would be simulated fully.

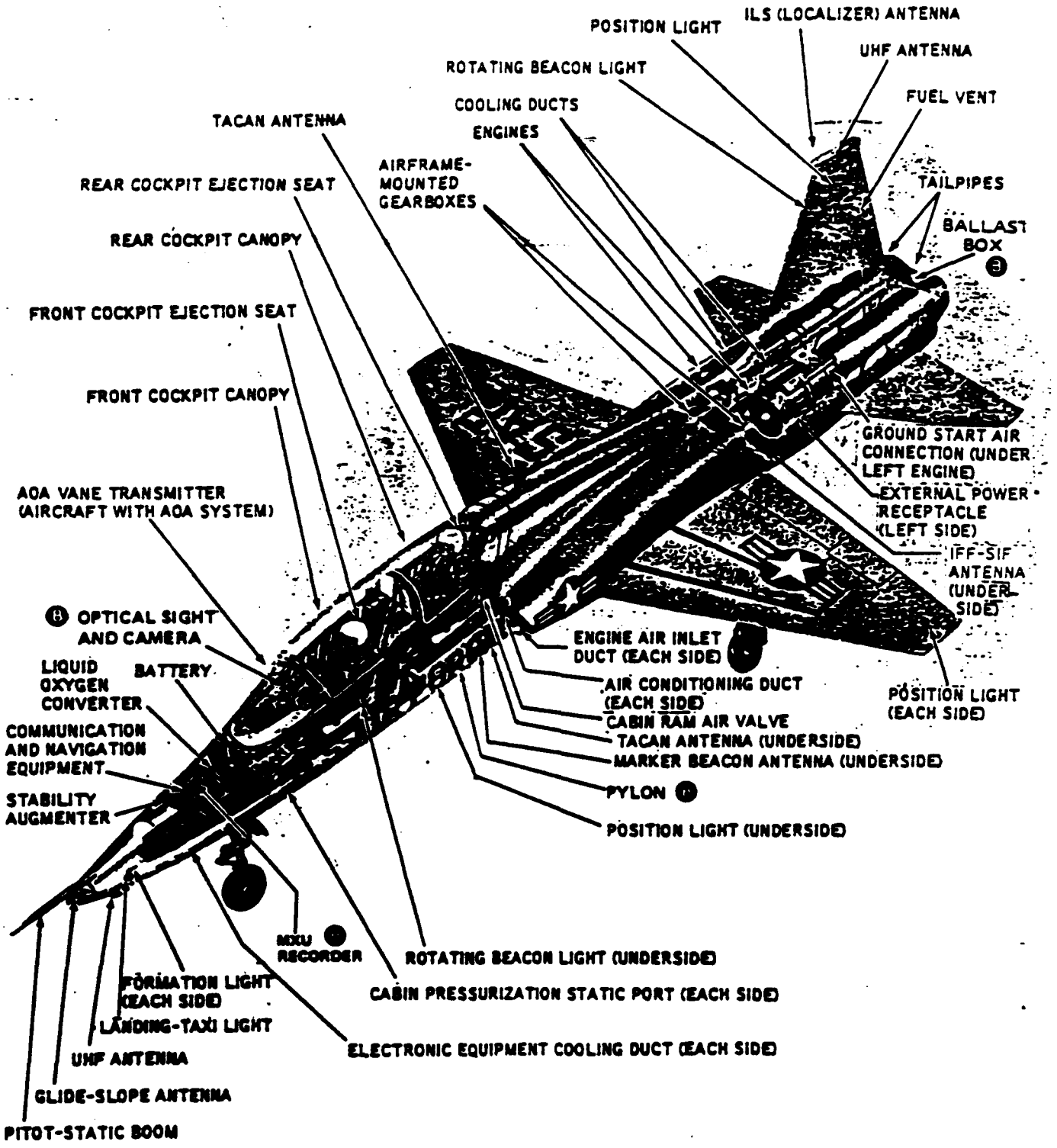


Figure 20. Top-level T-38 Aircraft View

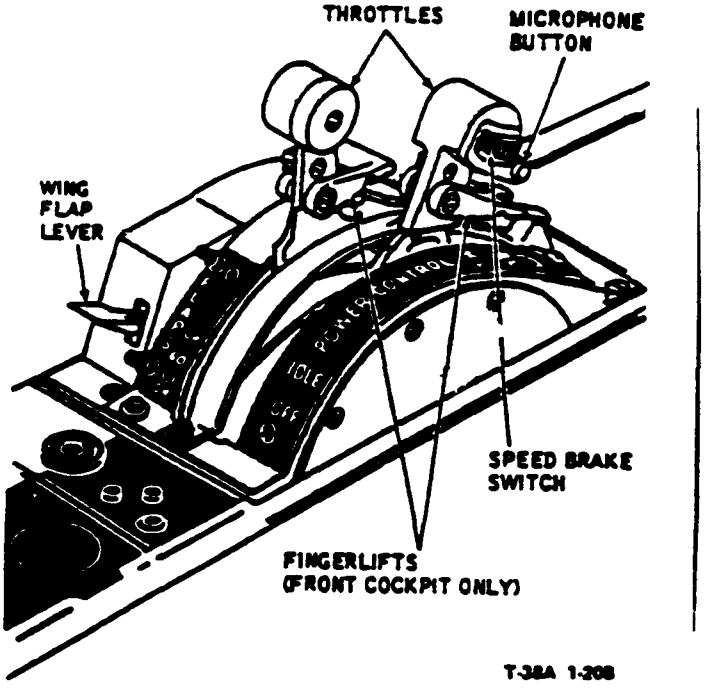


Figure 21. T-38 Throttle

Functional Equipment Model

Five figures are planned to represent the engine-starting components of the aircraft *schematically*, so the learner will be able to see how the system is organized internally and how it reacts as various operations are performed. These five views consist of one detailed schematic plus four versions of increasing simplification. Figure 22 shows the sketch of the most detailed view (screen figures are called *scenes*).

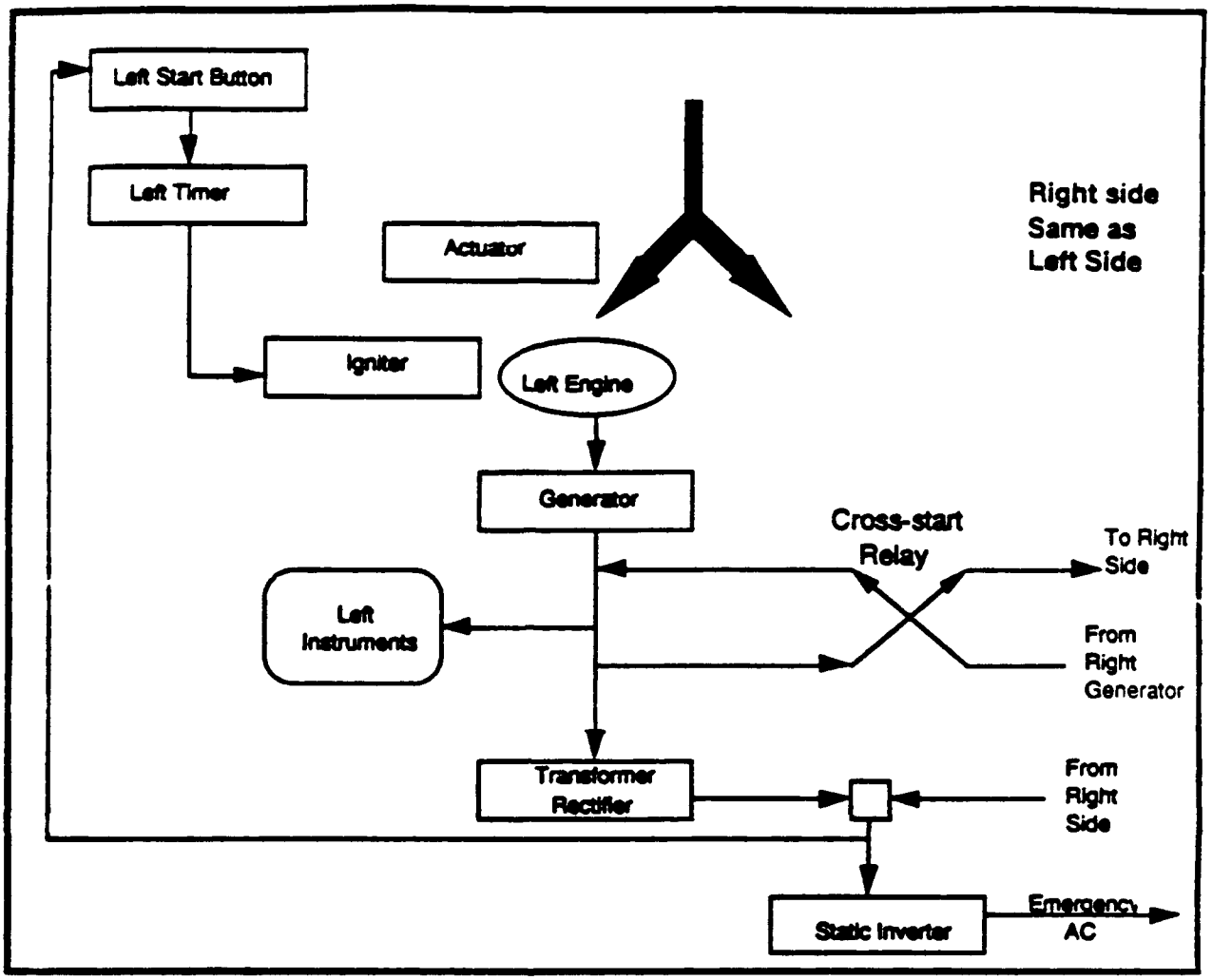


Figure 22. Detailed Schematic Scene - Preliminary Sketch

Produce objects that have not been previously created (24 hours SME)

The SME looks over the sections of the RAPIDS object library, a portion of which is shown in Figure 23, and finds the objects that can be used. Some of these will require minor modifications to make them suit this application exactly.

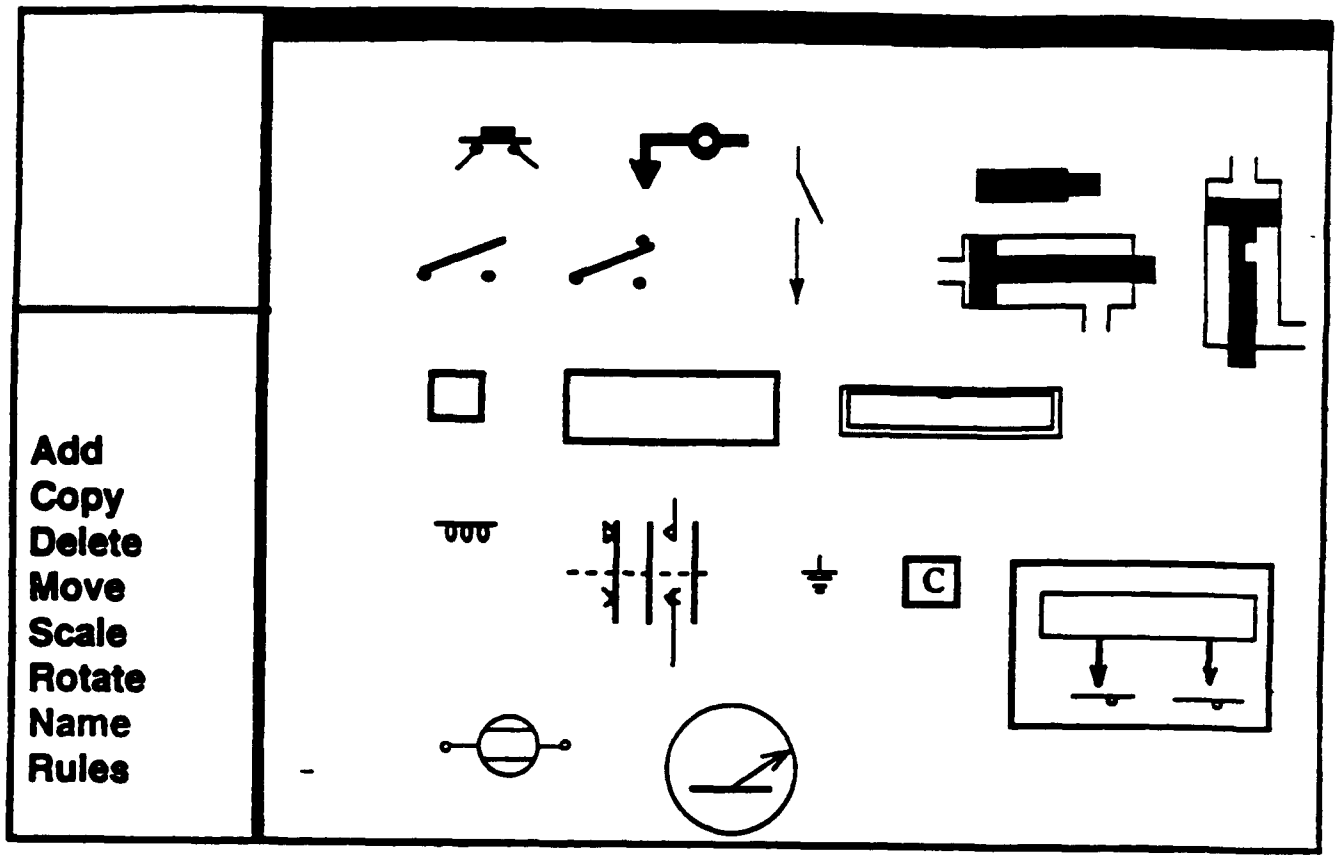


Figure 23. A Portion of the RAPIDS Object Library (functional section)

The SME will need to create about a dozen new schematic objects, and about eight new front panel parts, most of which are simple in appearance and function. One of the parts to be produced is the Diverter Valve, an object needed in the functional equipment model. Figure 24 shows it being drawn. This takes about 20 minutes.

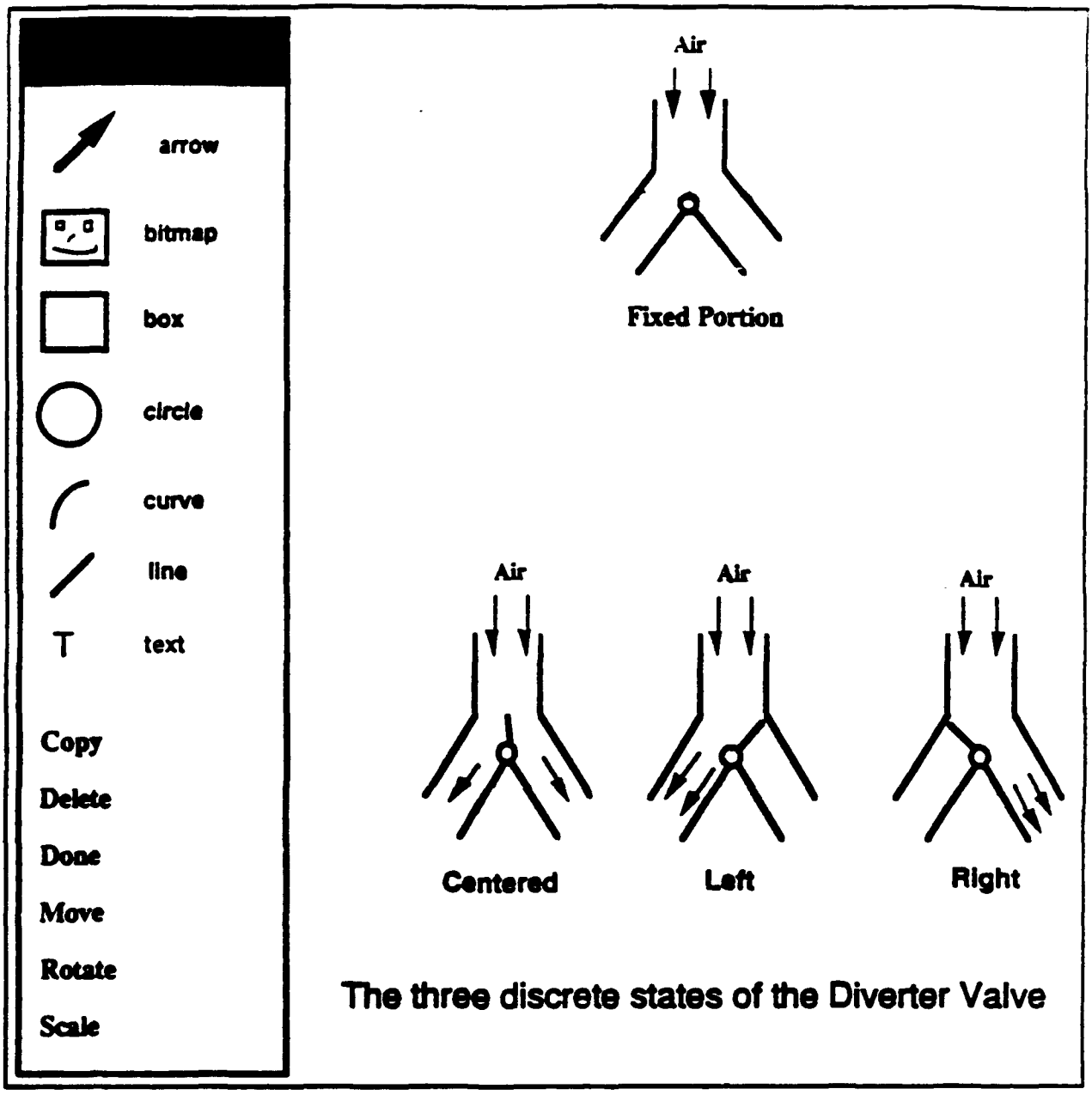


Figure 24. Drawing a New Functional Object

The behavior rules for this part are as follows. If there is more pull on the left side of the diverter valve vane (from the left actuator) than on the right side, then the vane is pulled to the left, and the air at the input is directed out the right channel. If more force is on the right side, then the vane is pulled to the right, and the air is directed out the left channel. If there is equal force on both sides, then the valve is centered, directing one-half of the air out each of the two

channels (which is insufficient to start either engine on the ground).

The SME sets the Diverter Valve into its Left state, then clicks on **Rules** (Figure 25), to supply the rules that govern the behavior of this part.

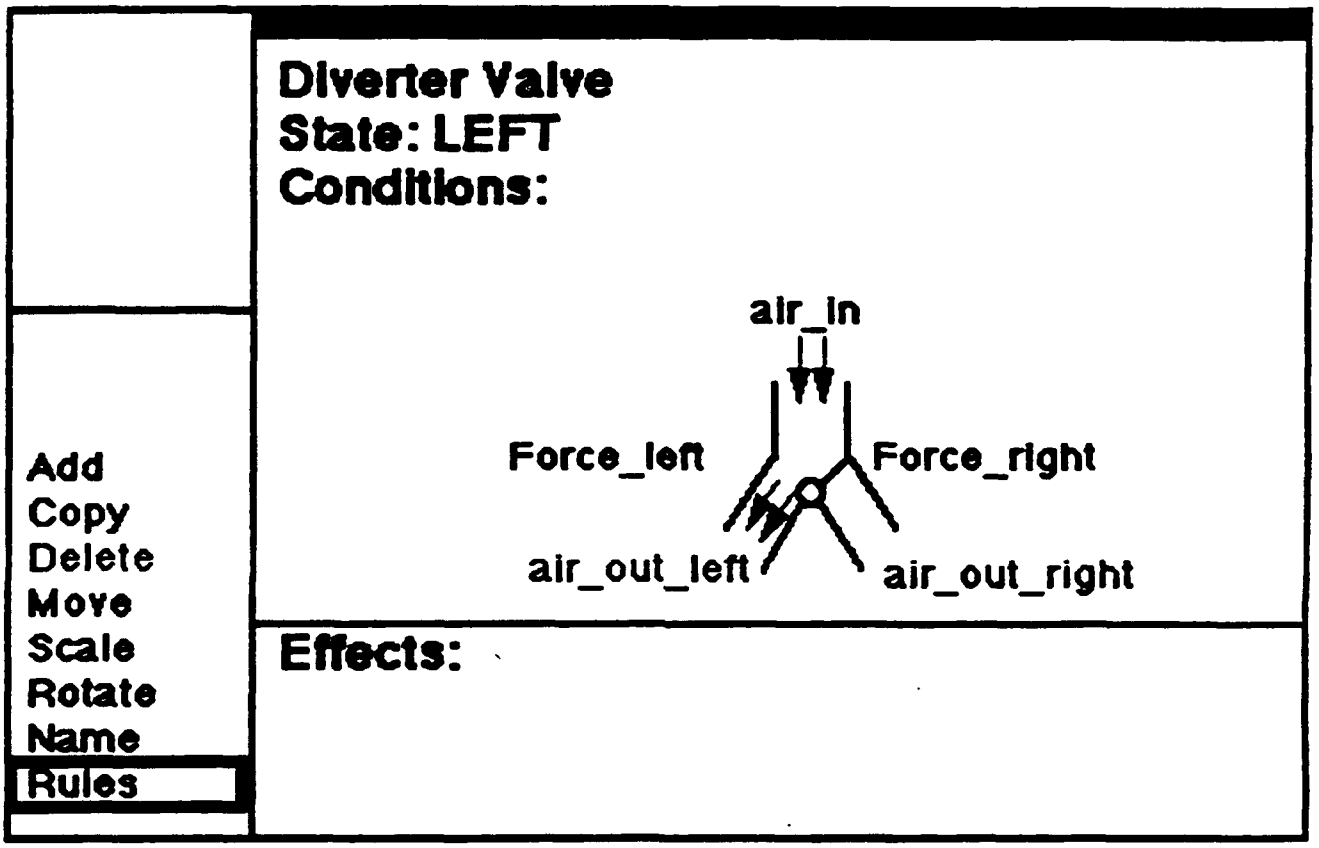


Figure 25. Starting to Define the Rules for the Diverter Valve

First, the author is prompted to specify the Conditions which produce this state (Figure 26), then the Effects that result when this state exists (Figure 27).

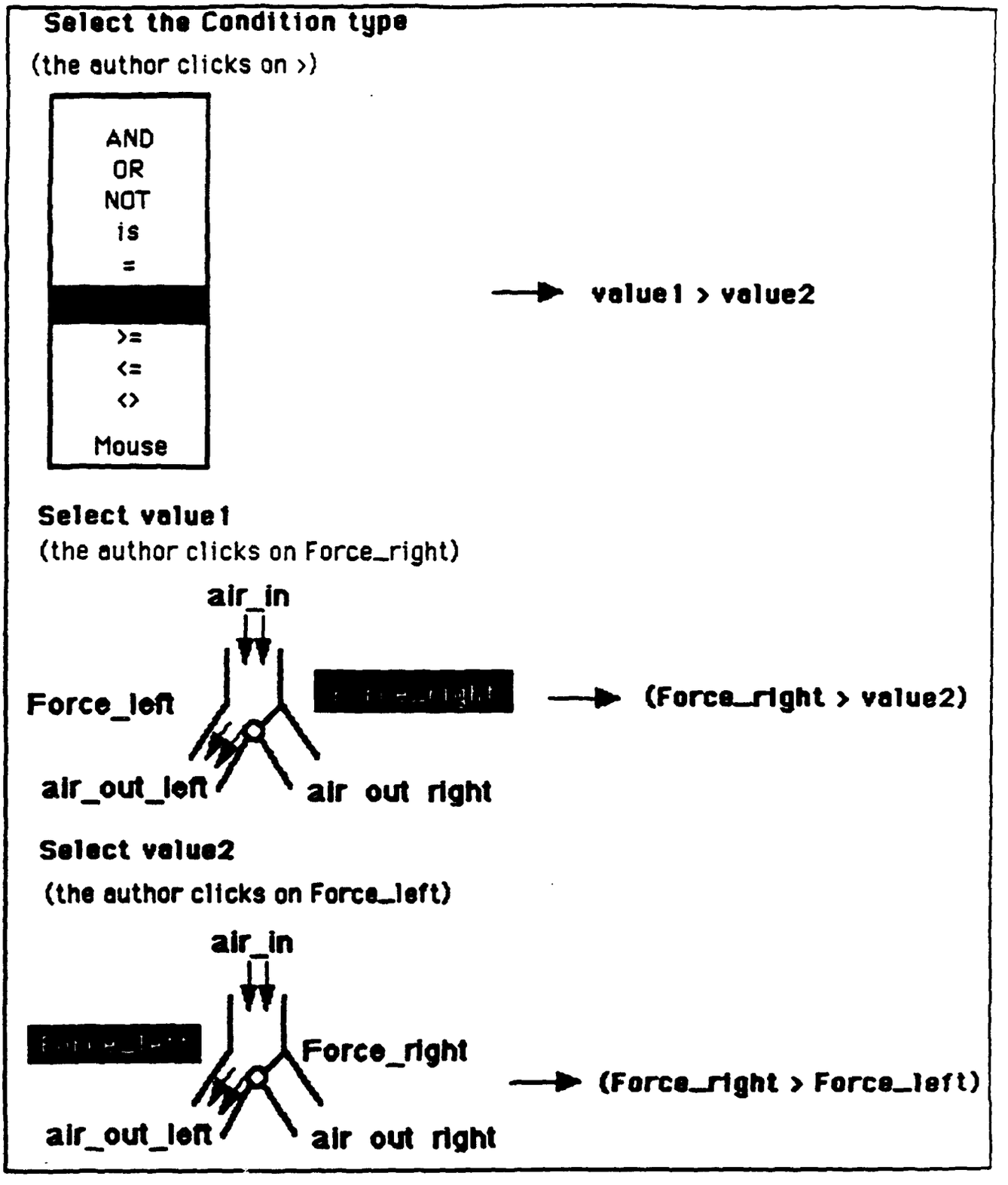


Figure 26. Authoring the Conditions for an Object State

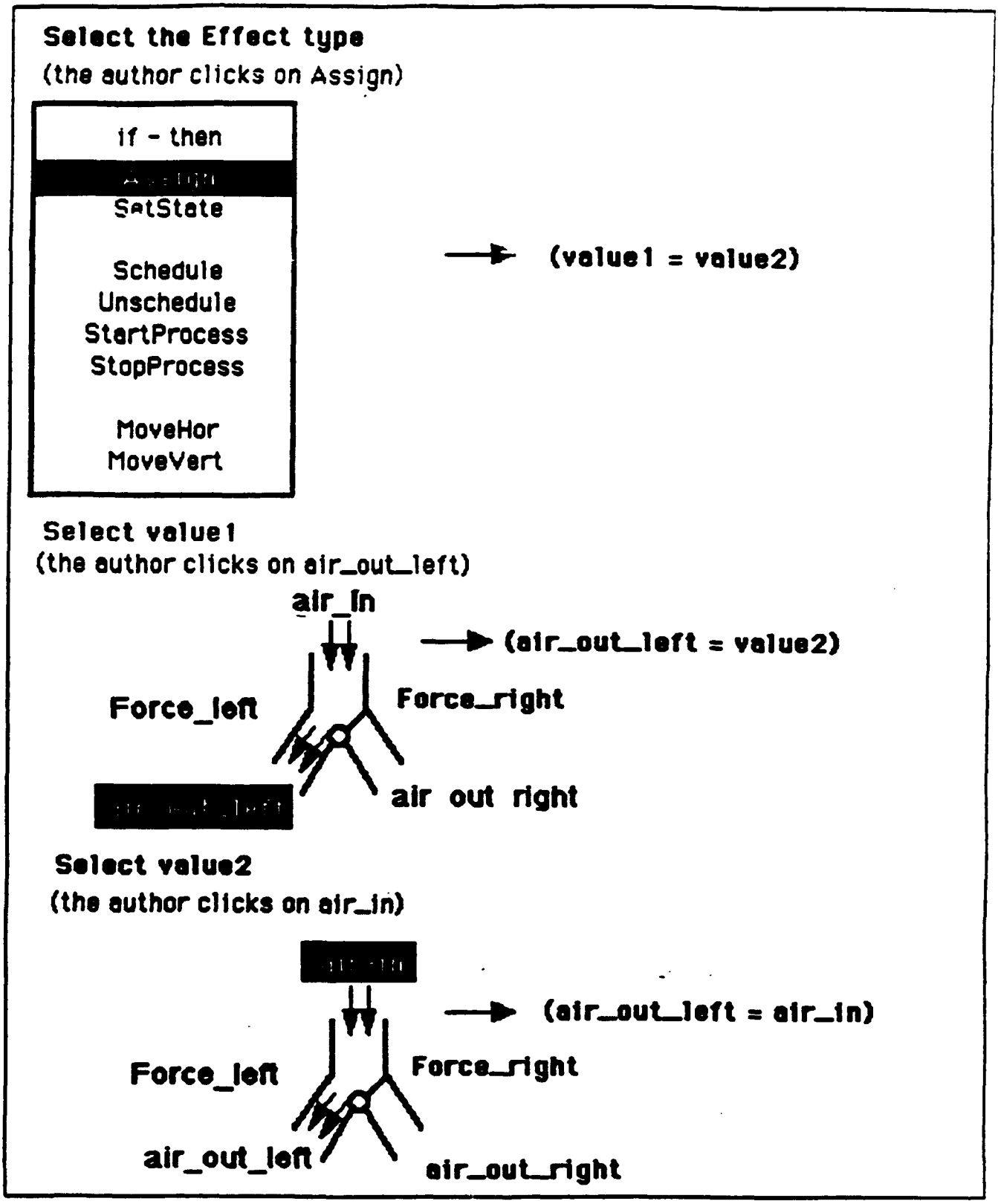


Figure 27. Authoring the Effects of an Object State

After entering the Conditions and Effects for the Left state, the screen appears as in Figure 28.

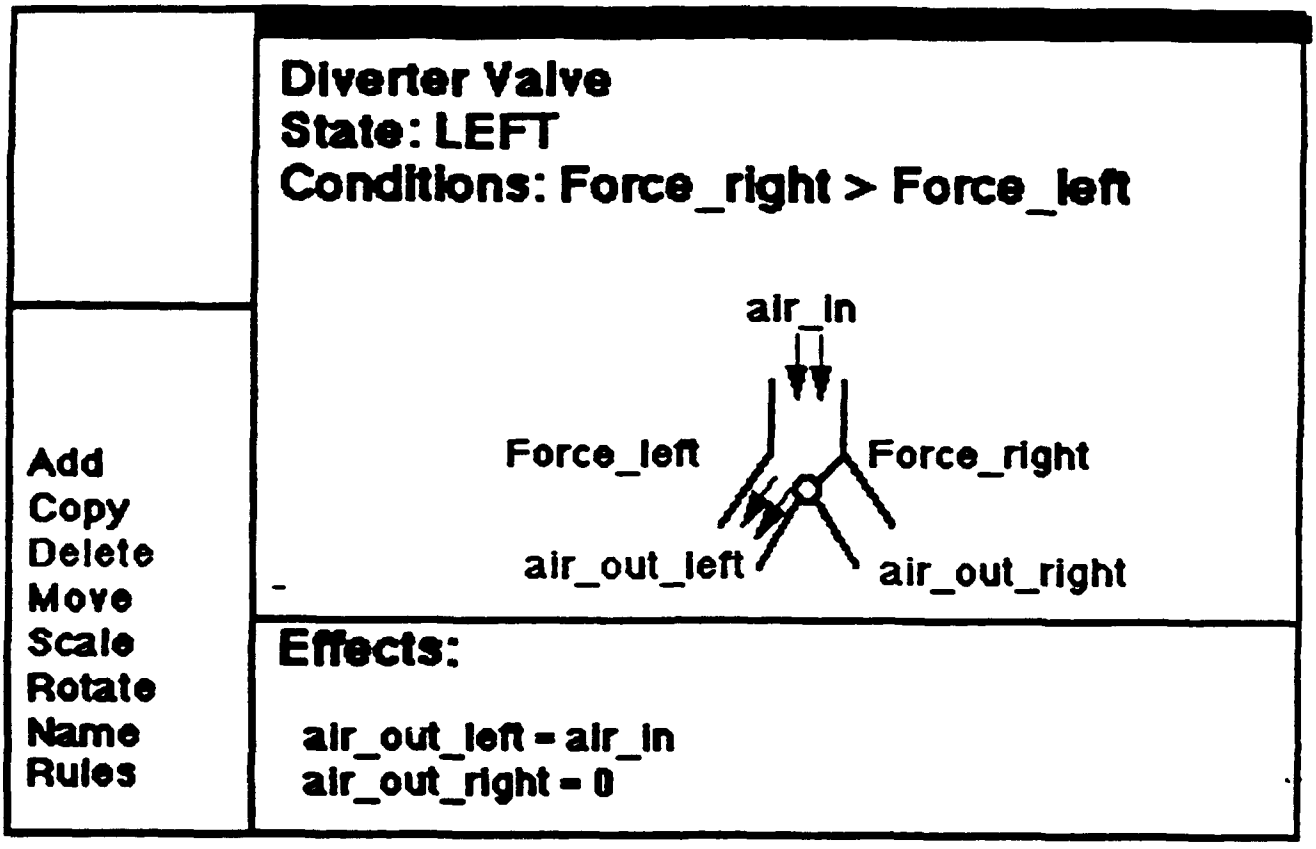


Figure 28. The Rules for One State of the Diverter Valve

The rules for the Right state are entered similarly.

Specify abnormal behaviors of objects (8 hours SME)

Because the instruction will simulate various failure conditions, it is necessary to add rules of behavior that components follow when they are faulty. Most of these are simple statements that various outputs are abnormal or missing. Any number of different failure modes may be defined for any object. The SME specifies two or three failure conditions for each of 25 objects, resulting in about 75 single failure conditions. RAPIDS can automatically simulate any combination of multiple failures, from these single-fault behavior rules.

Figure 29 shows the rules for a diverter valve that has become obstructed, thereby preventing the flow of air to either engine (the graphics representing a failed part are not shown during diagnostic practice, as they would reveal the problem).

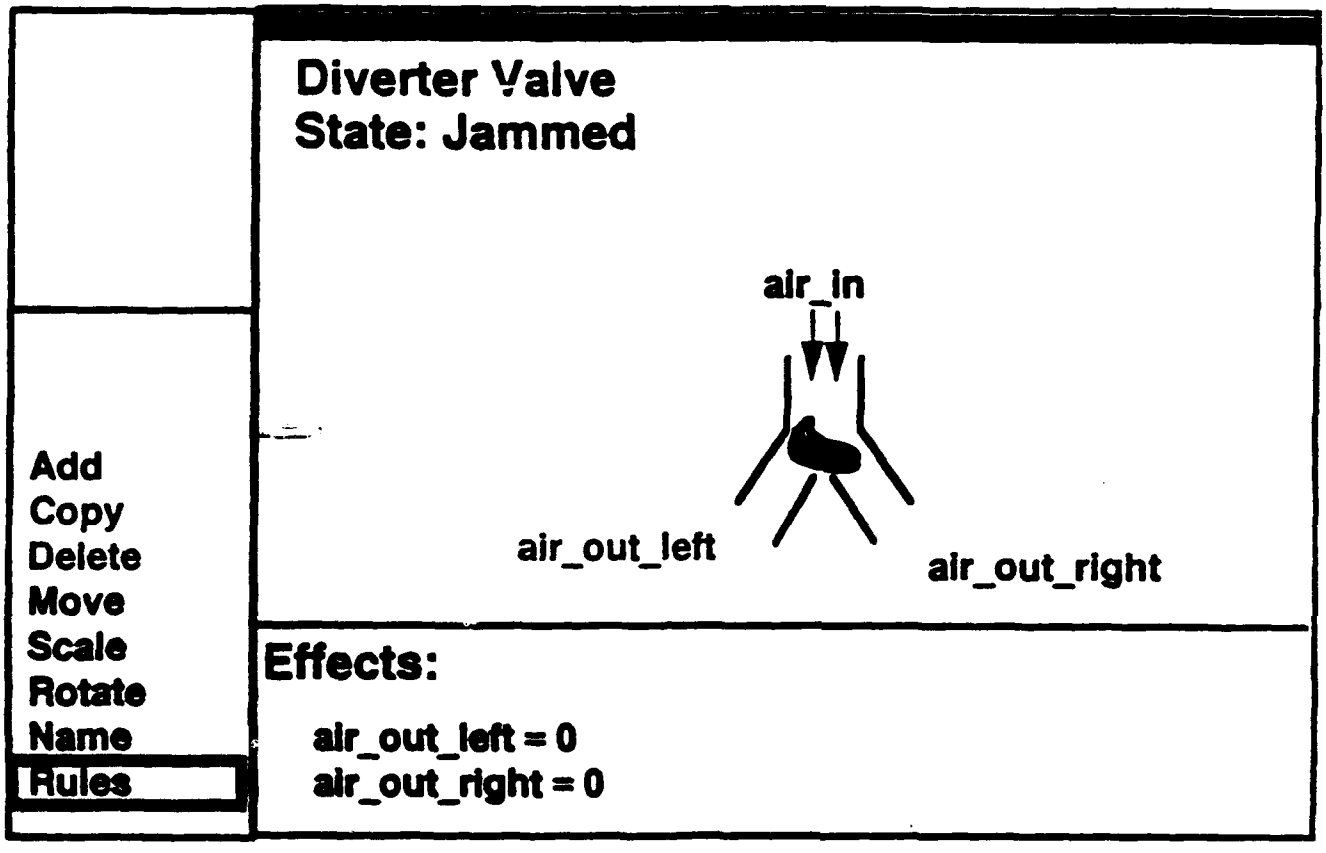


Figure 29. The Rules for One Failure Mode of the Diverter Valve

Construct the detailed scenes (24 hours SME)

A few of the equipment-arrangement scenes will only be used to train the learner in finding major sections of the T-38, and consequently have no dynamic controls or indicators. These scenes were created by scanning in figures from the Flight Manual. More detailed views are accessed via these high-level scenes.

The remaining scenes are created by selecting objects from the RAPIDS object library and positioning them on the screen. Figures 30, 31, and 32 illustrate three of the physical equipment scenes produced. These figures will be used to instruct

recognition and location of components in the T-38 as well as maintenance skills and procedures.

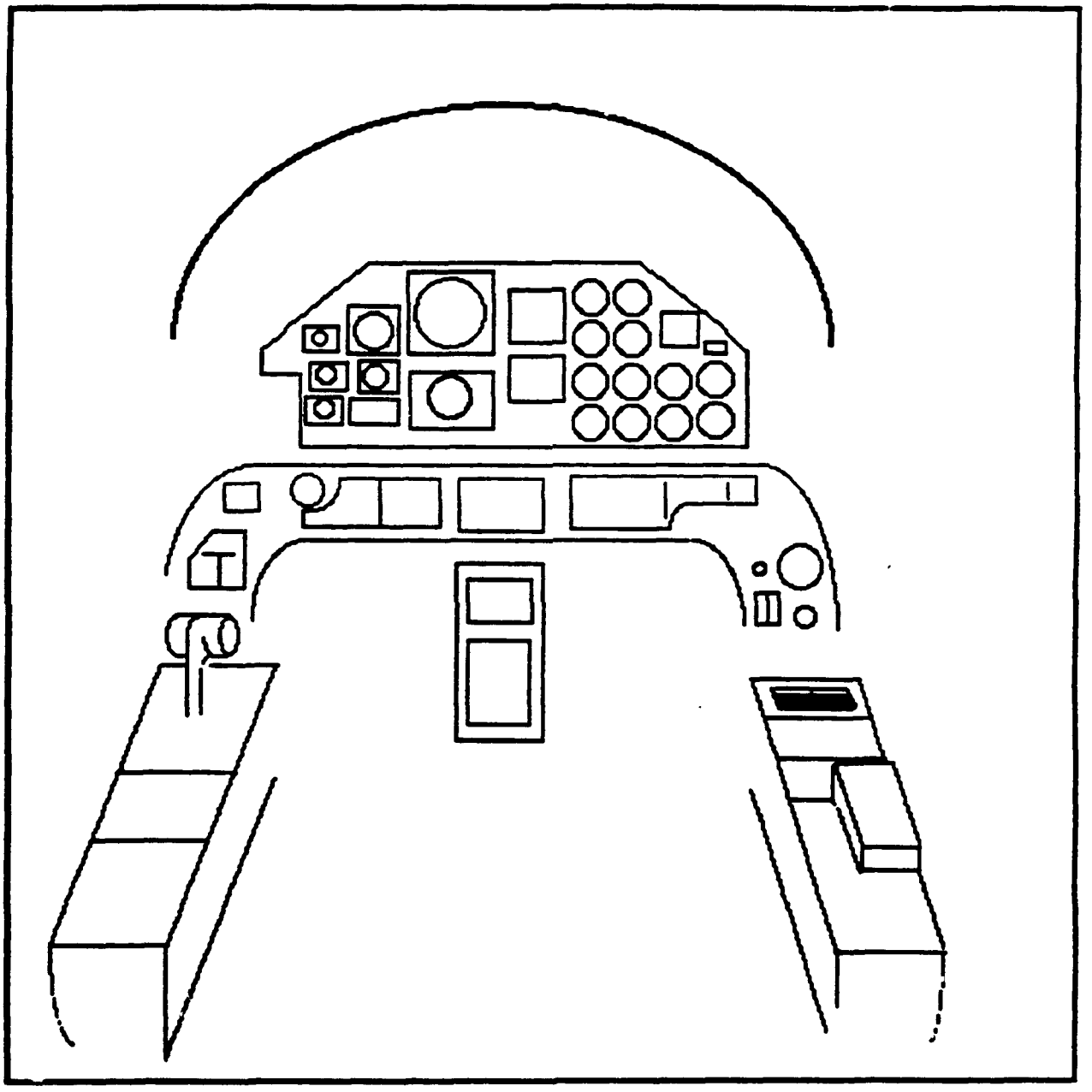


Figure 30. Cockpit Arrangement - Front

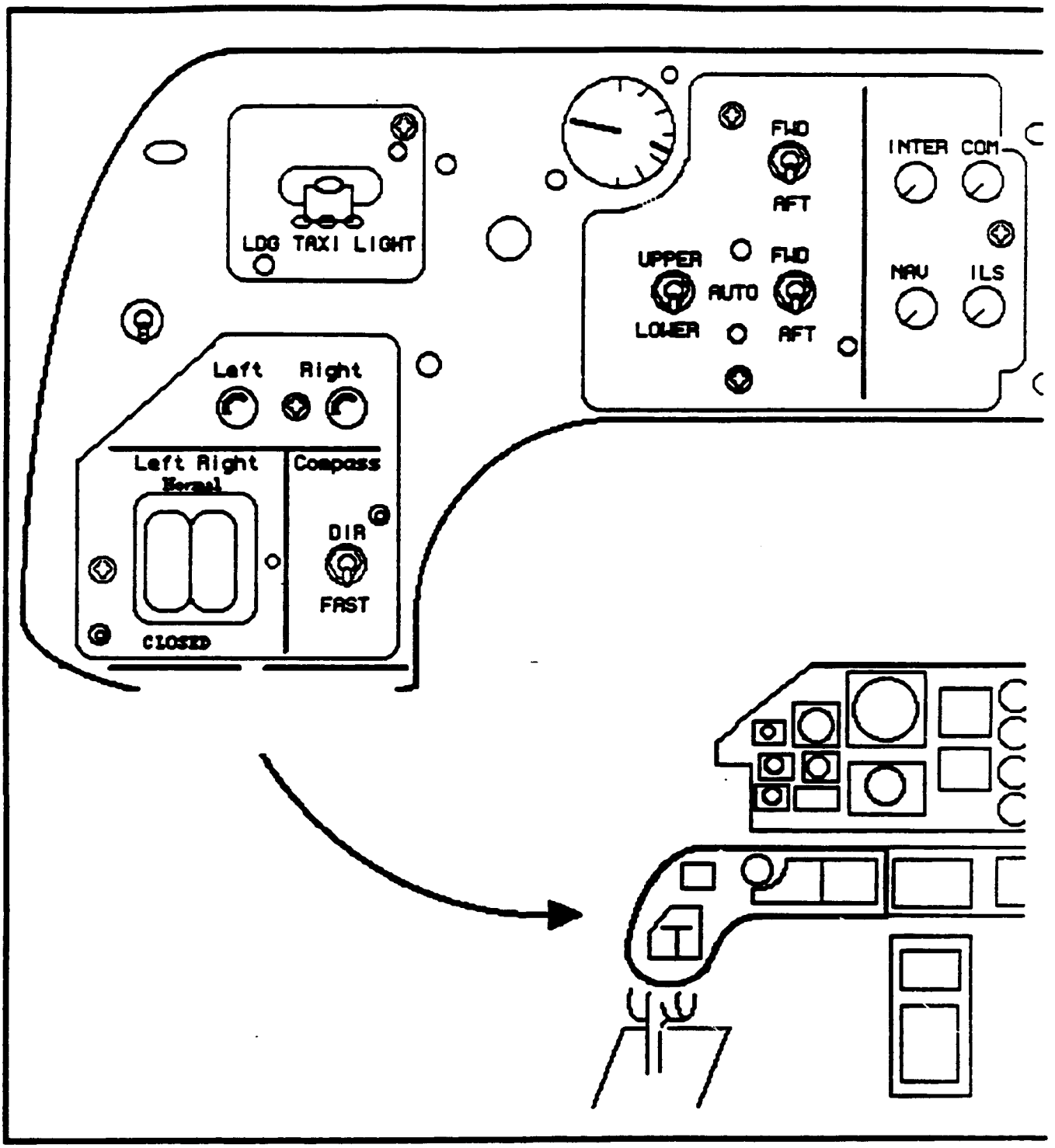


Figure 31. Left Subpanel

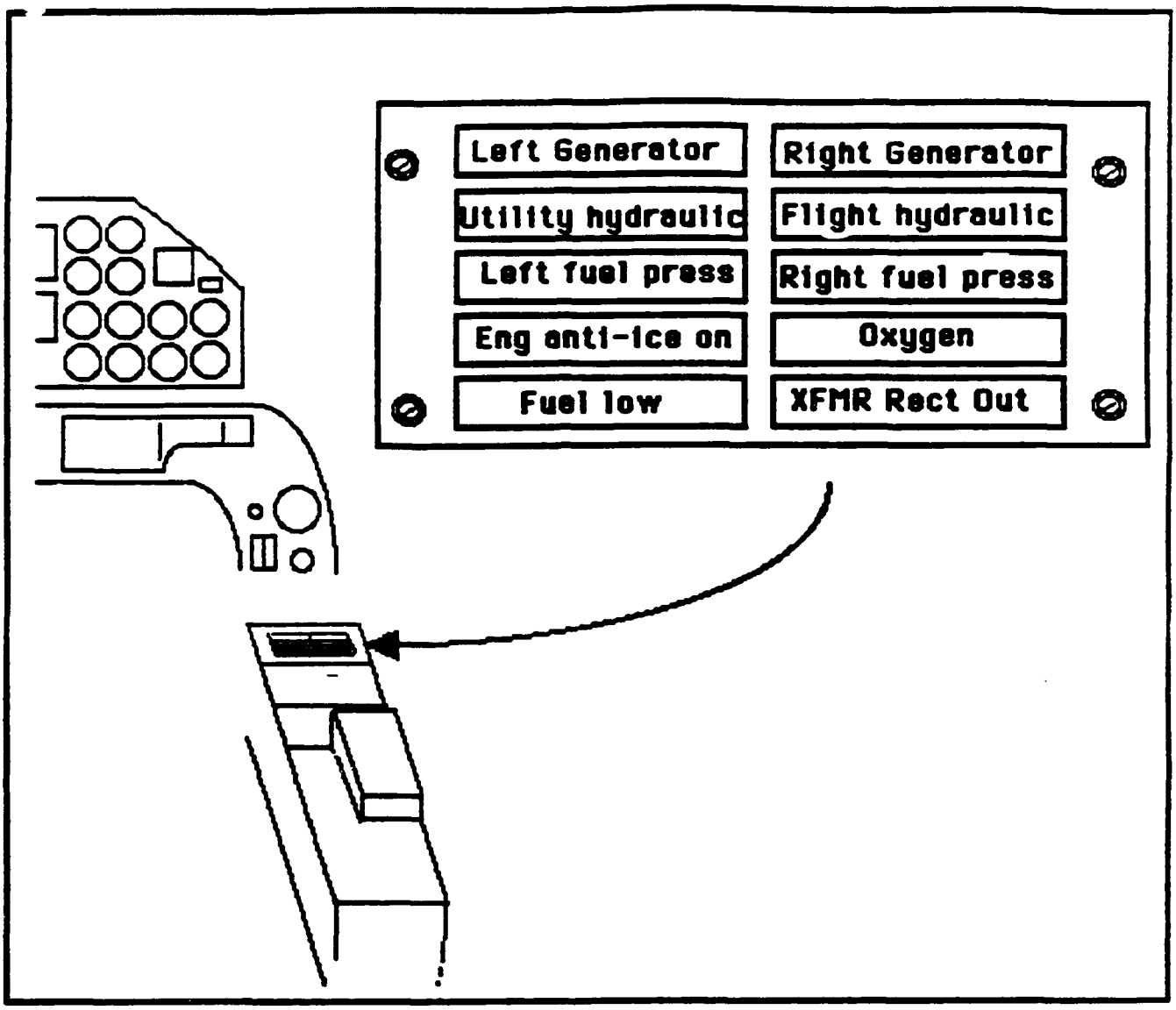


Figure 32. Caution Light Panel

The detailed *schematic* scene, showing all the components involved in engine starting, is produced from *functional* objects, as shown in Figure 33.

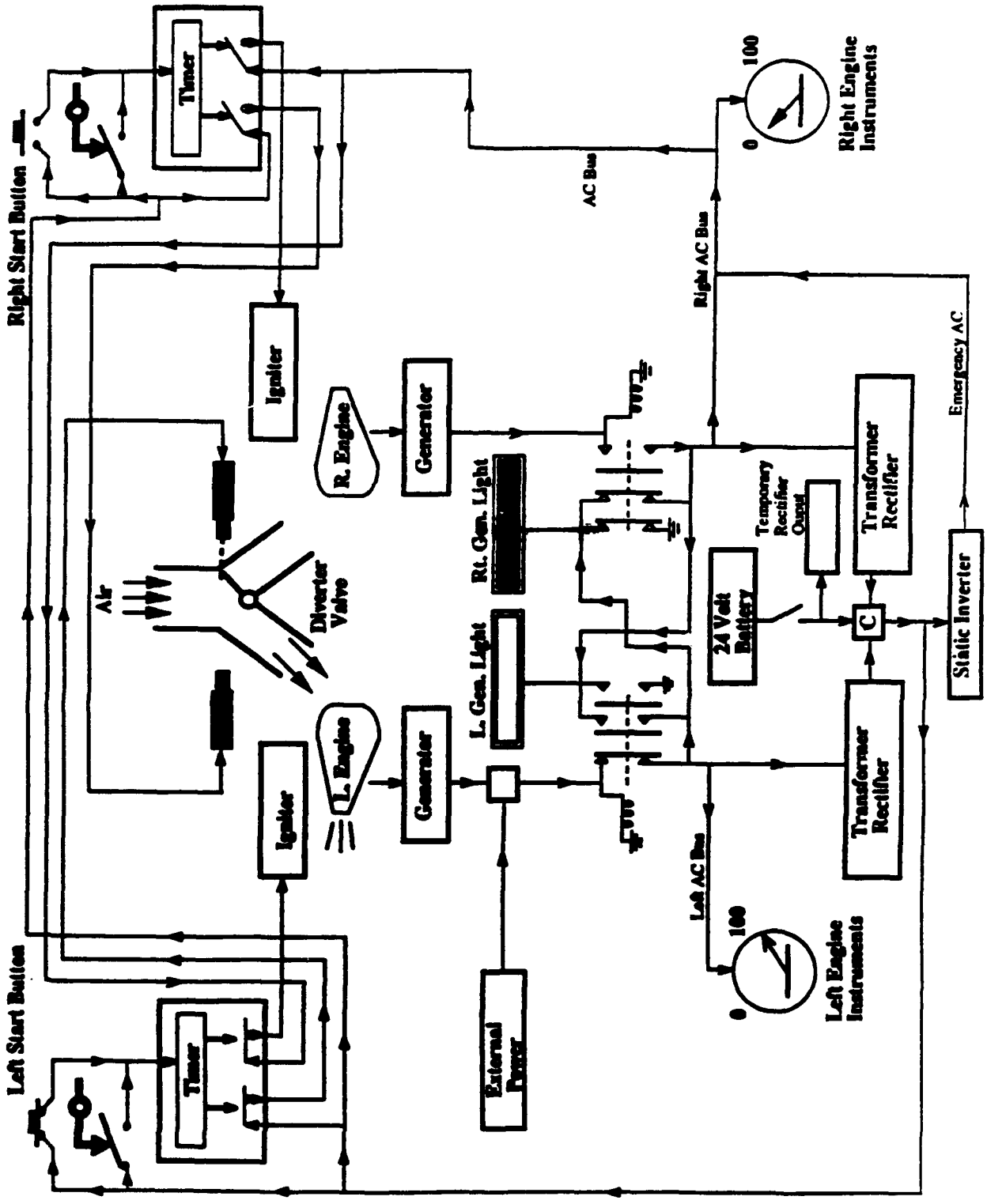


Figure 33. The Detailed Schematic Scene

The SME uses the functional equipment model to indicate how the various components affect one another. In Figure 34 the SME has selected two objects, and has drawn a line from the output of the Left Actuator to one of the inputs of the Diverter Valve. This line specifies that the Pull of the Left Actuator is the force felt by the Diverter Valve on the left side. After these *connections* are specified, the functional model can be operated.

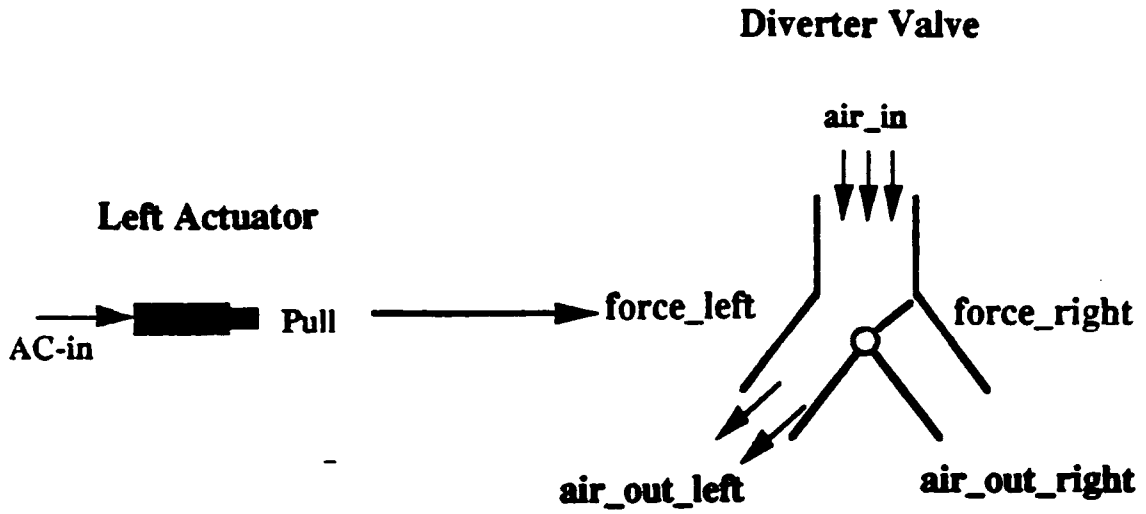


Figure 34. Specifying the Inter-Object Effects

Debug the functional equipment model (4 hours SME)

Once the detailed schematic scene is completed, the SME operates it, to test it for correct operation. For some reason the right engine won't start when the correct start procedure is conducted. The SME knows that the objects used in the right side are defined properly, since the same objects are used for the left side. The problem might lie in the definition of the diverter valve (the right-side), or the connections between objects on the right side. The SME uses RAPIDS's debug tools to determine the values of the inputs on the engine (Figure 35).

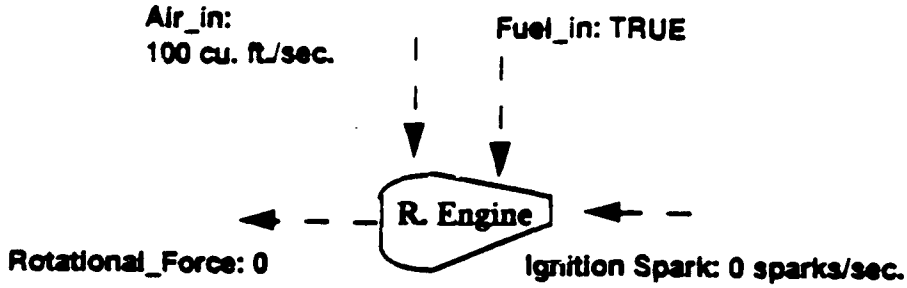


Figure 35. Displaying the Values of an Object's Attributes

This shows that the engine is not receiving ignition. The SME finds that he failed to connect the Right Igniter to the Right Engine. The SME fixes this, and the right engine starts correctly.

A second error is found: the engines won't start using the battery. The SME first selects the battery and sees a good output, then traces the outputs until finding a bad output from the Static Inverter. This leads the SME to discover and correct an error in the behavior rules for the Static Inverter which was just created.

Construct the four additional functional views (1 hour SME)

The SME makes a copy of the detailed (on-ground start) schematic scene, then removes the components involved in on-ground start from this copy. The resulting diagram depicts all the components involved in making an emergency start (Figure 36). This process is repeated, three more times, each time removing parts from the previous stage, to form successively simplified representations. The five schematic representations will be used to explain the following functions.

On-ground start (Figure 33)
Emergency start, no engines running (Figure 36)
In-air start, 1 engine running (Figure 37)
Cross-over system (Figure 38)
Generation of AC power (Figure 39)

The five schematics will be presented to the learner in order of increasing complexity during the instruction on theory of operation.

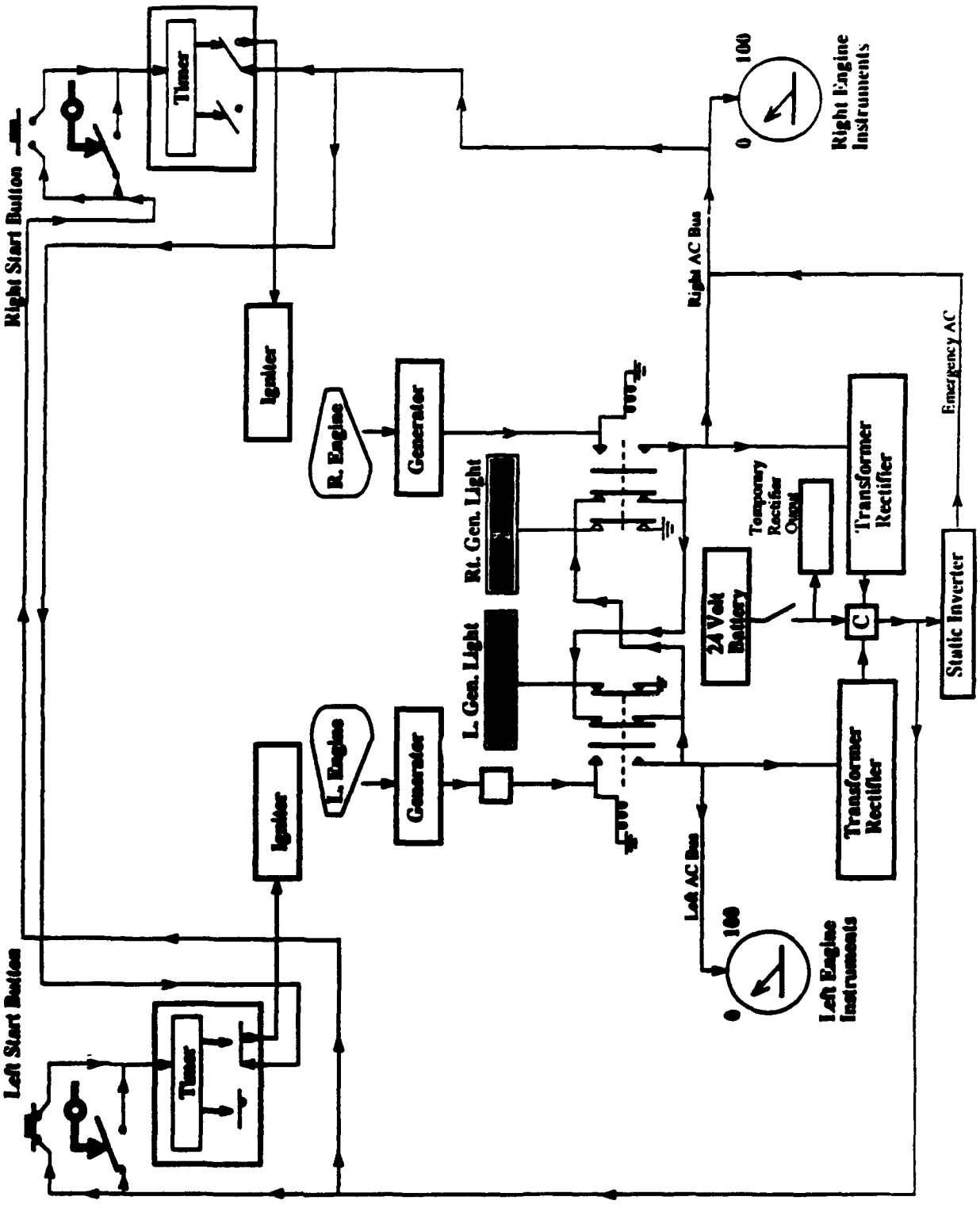


Figure 36. T-38 Emergency Start Functions

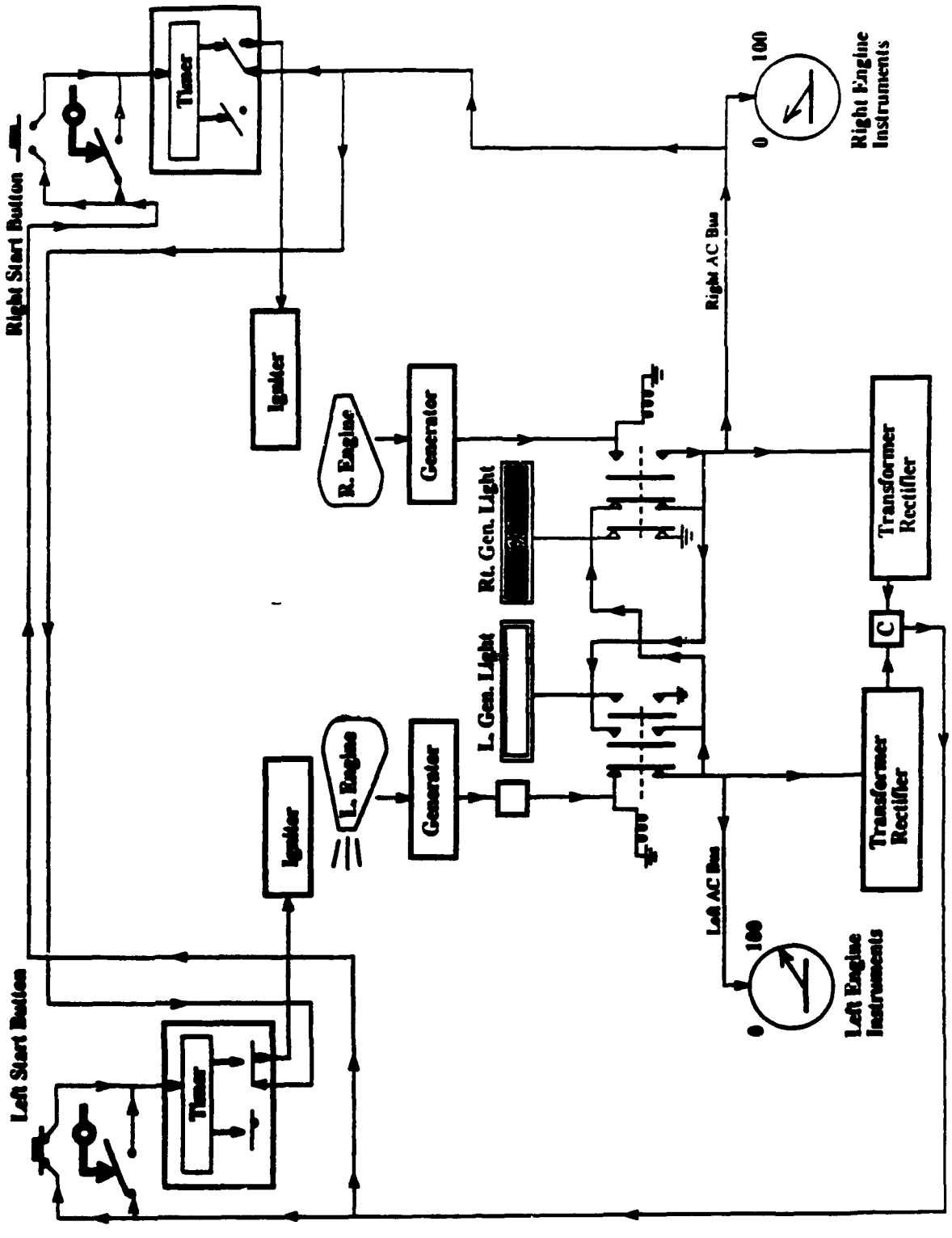


Figure 37. T-38 In-Air Start Functions

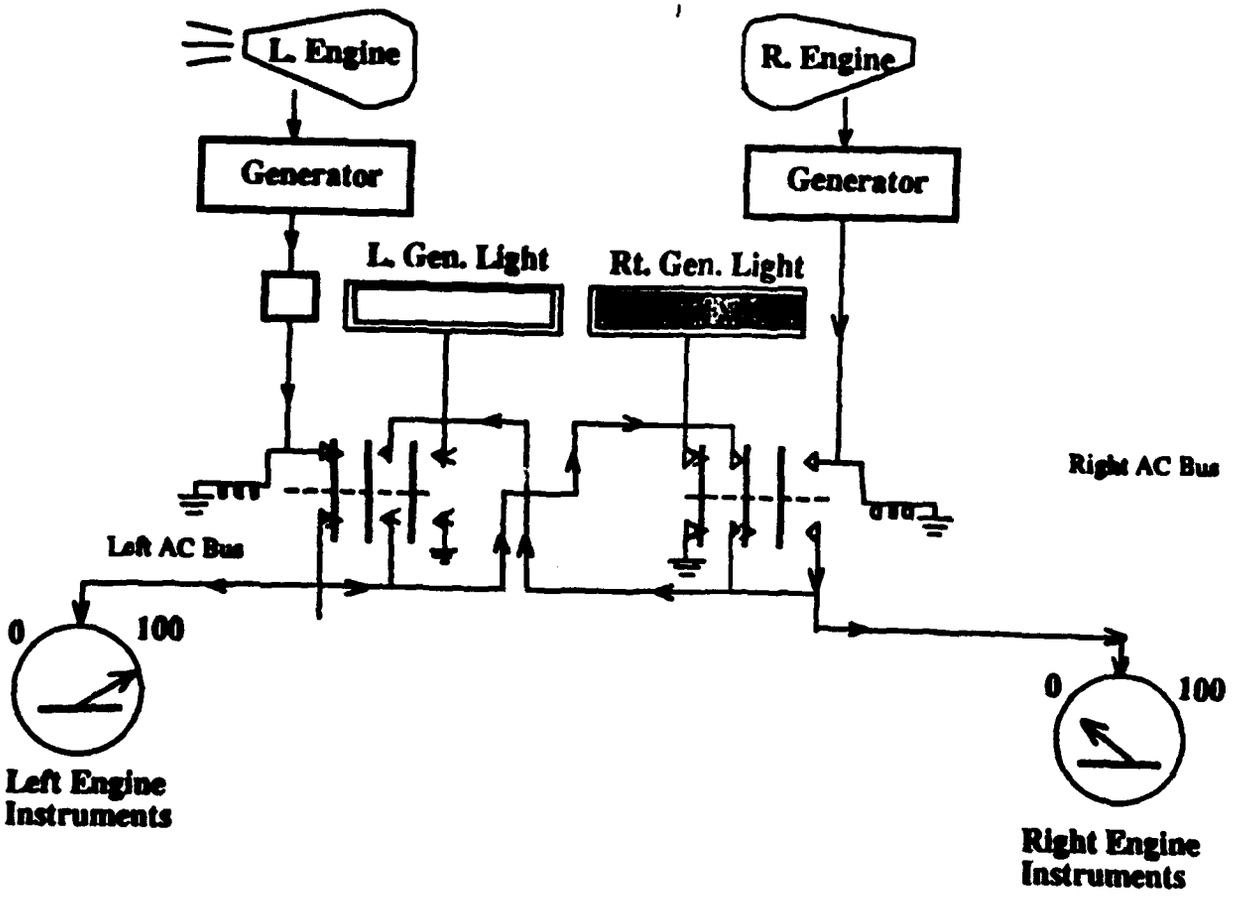


Figure 38. T-38 Cross-over System Functions

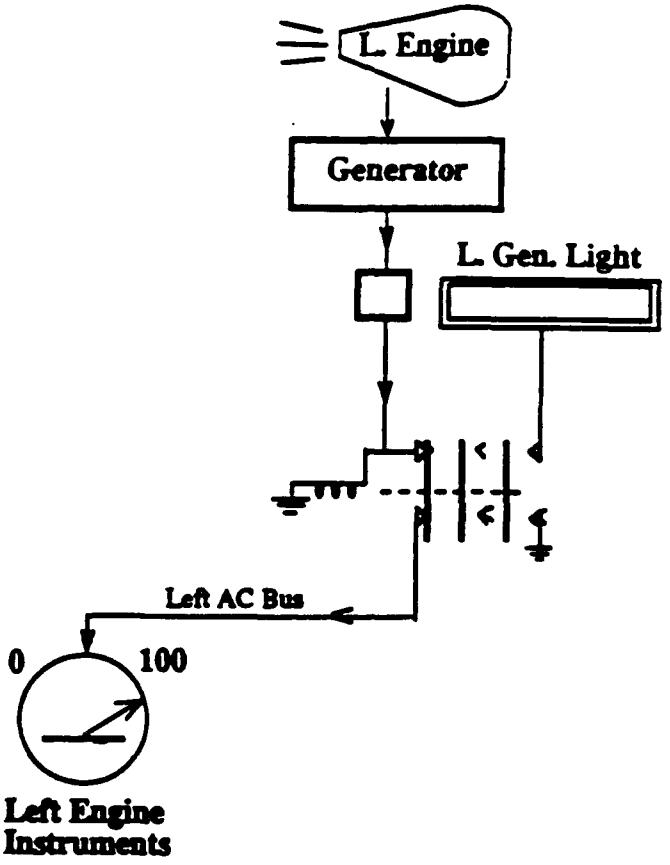


Figure 39. T-38 AC Power Generation Functions

Link the physical equipment model to the functional model (2 hours SME)

The SME indicates, via the mouse, which physical object corresponds with each functional object. By doing this, all the physical models become operational, as they are driven by the functional model. Furthermore, actions made upon the physical model can be viewed functionally, and *vice versa*.

Execute the fault-effect analysis routine (1 hour SME)

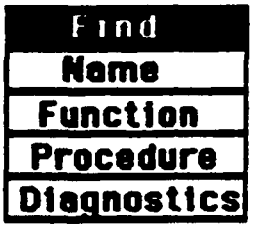
Many of the diagnostic training exercises are produced automatically (i.e., without authoring). This is made possible by executing a preprocessor program that systematically steps through each defined failure condition of each object, running the simulation and determining what abnormalities result. These fault-effect data are automatically saved for efficient use by the diagnostic instruction routines.

Producing the Instruction

The SME operates the RAPIDS T-38 models and produces auxiliary instructional expositions to provide the instruction called out in the instructional plan. *Most of the diagnostic instruction is produced automatically, without authoring.*

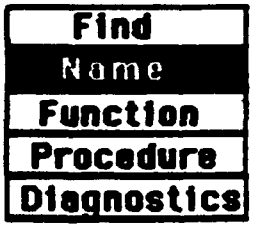
Produce the front panel location drill (0.1 hour SME)

The SME selects the **Find** exercise type, and selects the physical scenes as the scope of the exercise. RAPIDS will produce the instruction automatically.



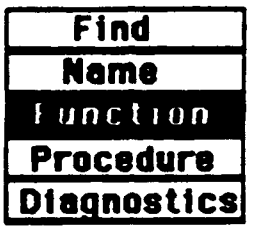
Produce the front panel nomenclature drill (0.1 hour SME)

The SME selects the **Name** exercise type, and selects the physical scenes as the scope of the exercises. RAPIDS will produce the instruction automatically.



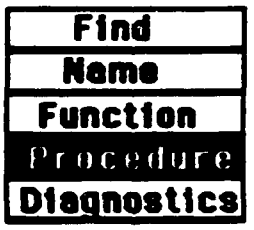
Produce the function recognition drill (1.0 hour SME)

The SME selects the **Function** exercise type, then keys in short functional descriptions of each component in the functional equipment model. RAPIDS will produce the instruction automatically (and will display the function of each component as it is identified in front panel orientation instruction).



Produce the Safety Procedures instruction (1 hour SME)

The SME selects the **Procedure** exercise type, then performs each step of the various safety procedures, adding text that explains the importance of each action and the implications of not following the procedures (see Section 4 for examples - the authoring appears virtually identical to the instructional delivery). In addition, the SME highlights areas that should be noticed at each step.



Produce the Theory of Operation instruction (4 hours SME)

Generation of AC Power

The SME selects the simplest schematic scene (Figure 39), then traces production of AC power from the running left engine, through the generator, the cross-start relay, and to the left instruments. Questions are interspersed that will require the student to respond.

The Cross-over System

The Cross-over scene is selected (Figure 39), and the SME shows how AC power is routed from either running engine to the other. The tracing is done by clicking the mouse on each section of the path, resulting in a highlighted route through the involved objects.

In-air Start Functions

The functions of the Igniter, timer, throttle, and start buttons are demonstrated in a condition with one engine running and the other failed in the air (Figure 37). The production of DC power and its routing to the start functions is explained.

Emergency Start Functions

The use of the 24-volt battery to produce the required DC to start is explained, along with the function of the static inverter for producing emergency AC power (Figure 36).

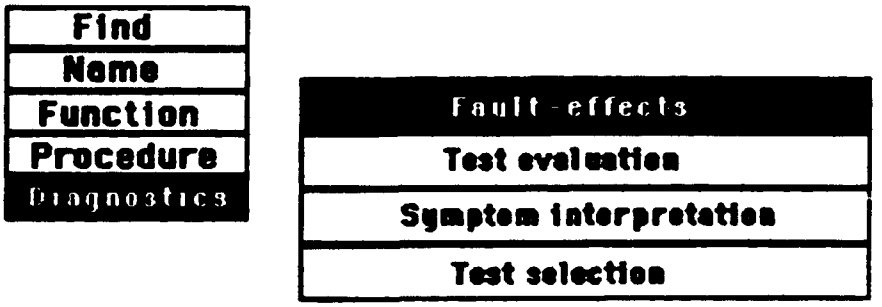
On-ground Start Functions

Starting on the ground is explained by showing the external power and externally-provided air supply. The function of the air diverter valve is demonstrated and explained (Figure 33).

Produce the Fault Diagnosis instruction (0.5 hr. SME)

Fault-effects

The SME selects the Fault-effects exercise type, a submenu of the Diagnostic exercises, and selects ALL to include all available faults. RAPIDS will produce the instruction automatically.



Test Evaluation

The SME selects the TEST-EVALUATION exercise type and selects ALL to include all indicators. RAPIDS will produce the instruction automatically.

Symptom Interpretation

The SME selects the SYMPTOM-INTERPRETATION exercise type and selects ALL to include all symptoms. RAPIDS will produce the instruction automatically.

Test Selection

The SME selects the TEST-SELECTION exercise type. RAPIDS will produce the instruction automatically.

Produce the Free-exploration instruction (0.1 hr. SME)

The SME enters some directions to the student, suggesting that he or she should insert failures of interest and explore the symptoms that result, comparing these to normal indications. RAPIDS will administer this free exploration mode automatically, including 1) responding to the learner's request to insert or remove various faults, 2) maintaining the equipment model as the learner works, and 3) providing access to the frame-based knowledge structure.

Produce the Fault Diagnosis Practice (0.5 hours SME)

The SME selects the PRACTICE-PROBLEM exercise type, then selects 20 failures to be presented. RAPIDS will administer the instruction automatically by inserting the selected failures (in random order if so requested) and assisting the learner in practicing fault isolation.

Produce the Safety Review (0.1 hour SME)

This unit of instruction employs the same unit that was produced for the original instruction in safety. The SME enters some instructions that advise the student that this is a review of previously learned safety procedures.

Sample Instruction Scenarios

The following are representative examples of the instruction received by the student. On the color screens that will be used in RIDES, highlighting will be easily seen by the student, thus the arrows shown in this walk-through will not be required to draw the student's attention.

Front Panel Orientation

RAPIDS drills the student in finding the various controls and indicators in the T-38 cockpit. A nomenclature drill is also presented, so the technician will acquire the terms used in the manuals.

The initial *presentation* mode, in which RAPIDS takes the student through the equipment, pointing out all the controls and indicators, appears as shown below.

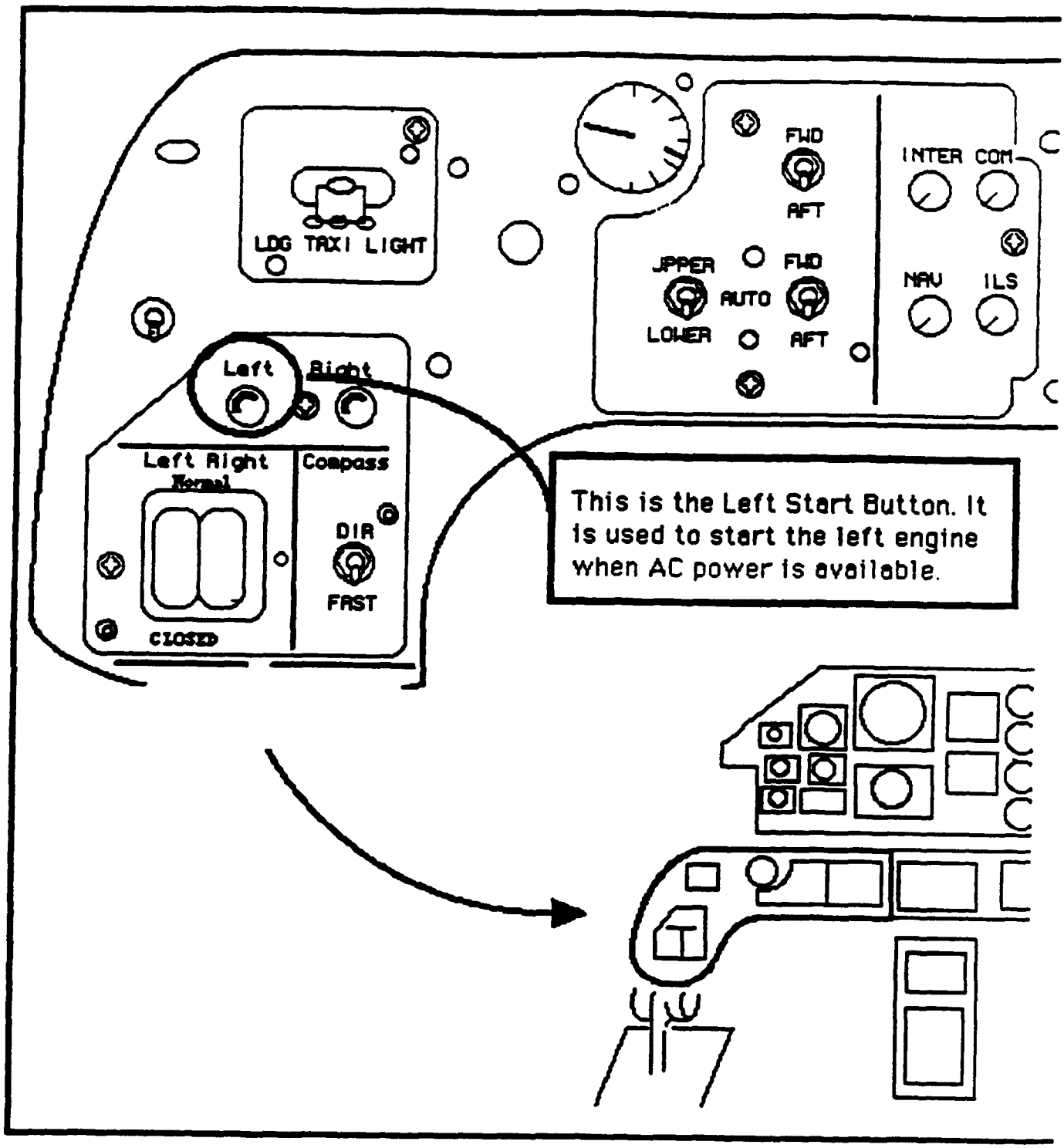


Figure 40. Front Panel Orientation (presentation mode)

In *drill* mode, the learner is shown the top-level view of the cockpit, and required to find individual front panel elements. This requires that the learner first go to the proper exploded view, by finding the correct subpanel.

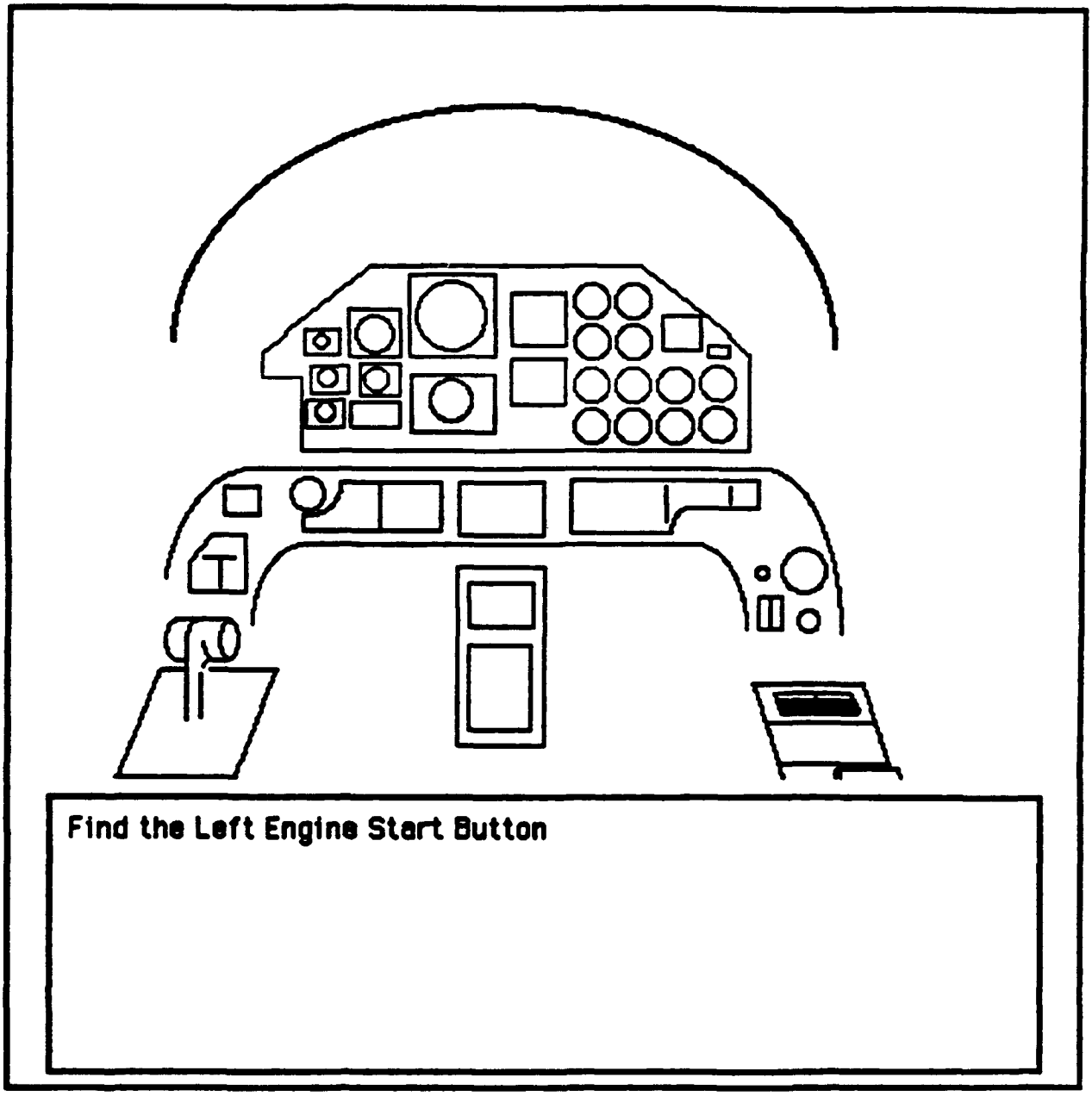


Figure 41. Front Panel Orientation (drill mode, initial view)

The learner first clicks on the Left Subpanel, bringing up the following view. Then, the learner selects the Left Start Button. If the learner has any problems, RAPIDS assists automatically.

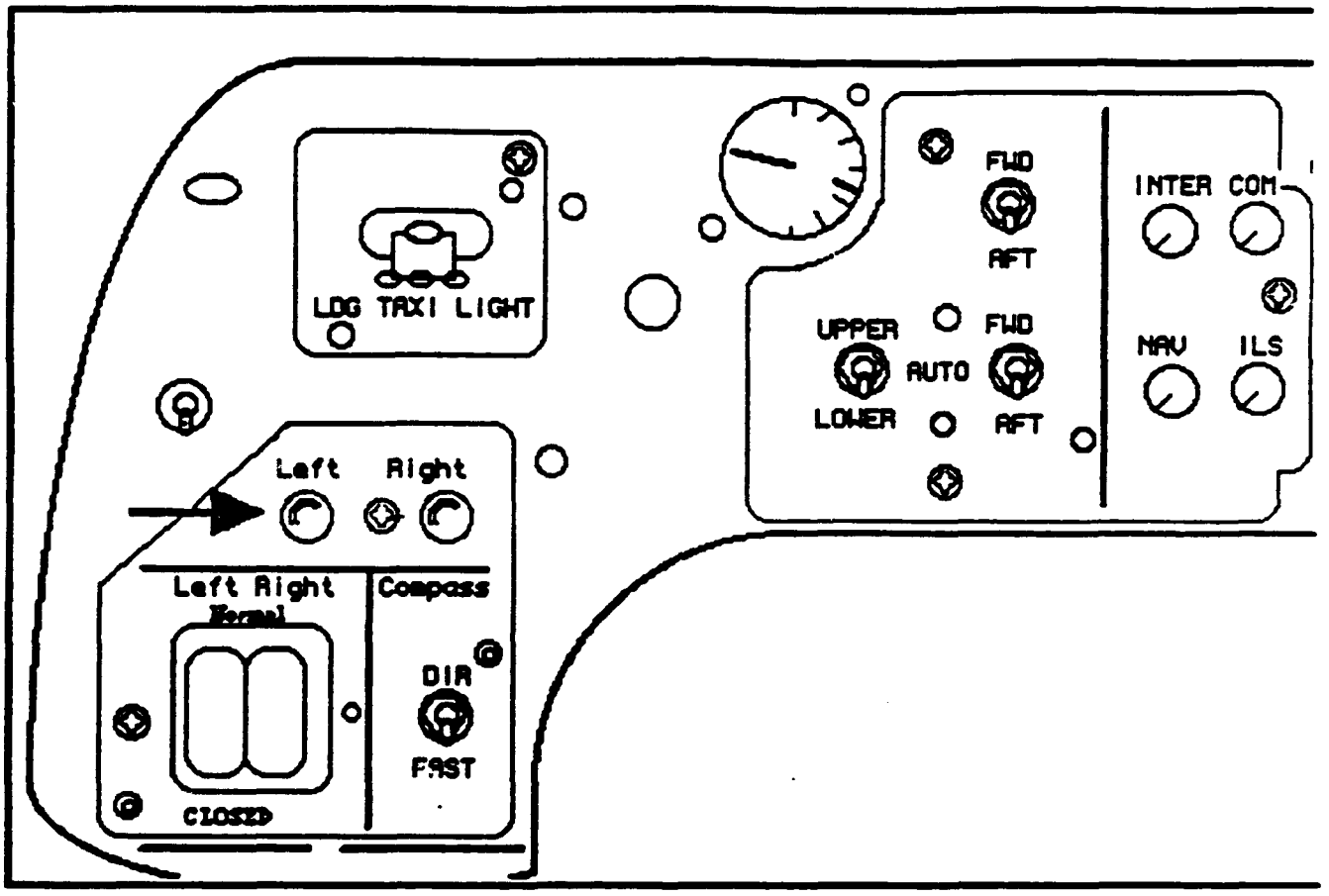


Figure 42. Front Panel Orientation (drill mode, final view)

Safety Procedures

The safety procedures and warnings entered by the SME are now played for the student, as shown in Figures 43 and 44.

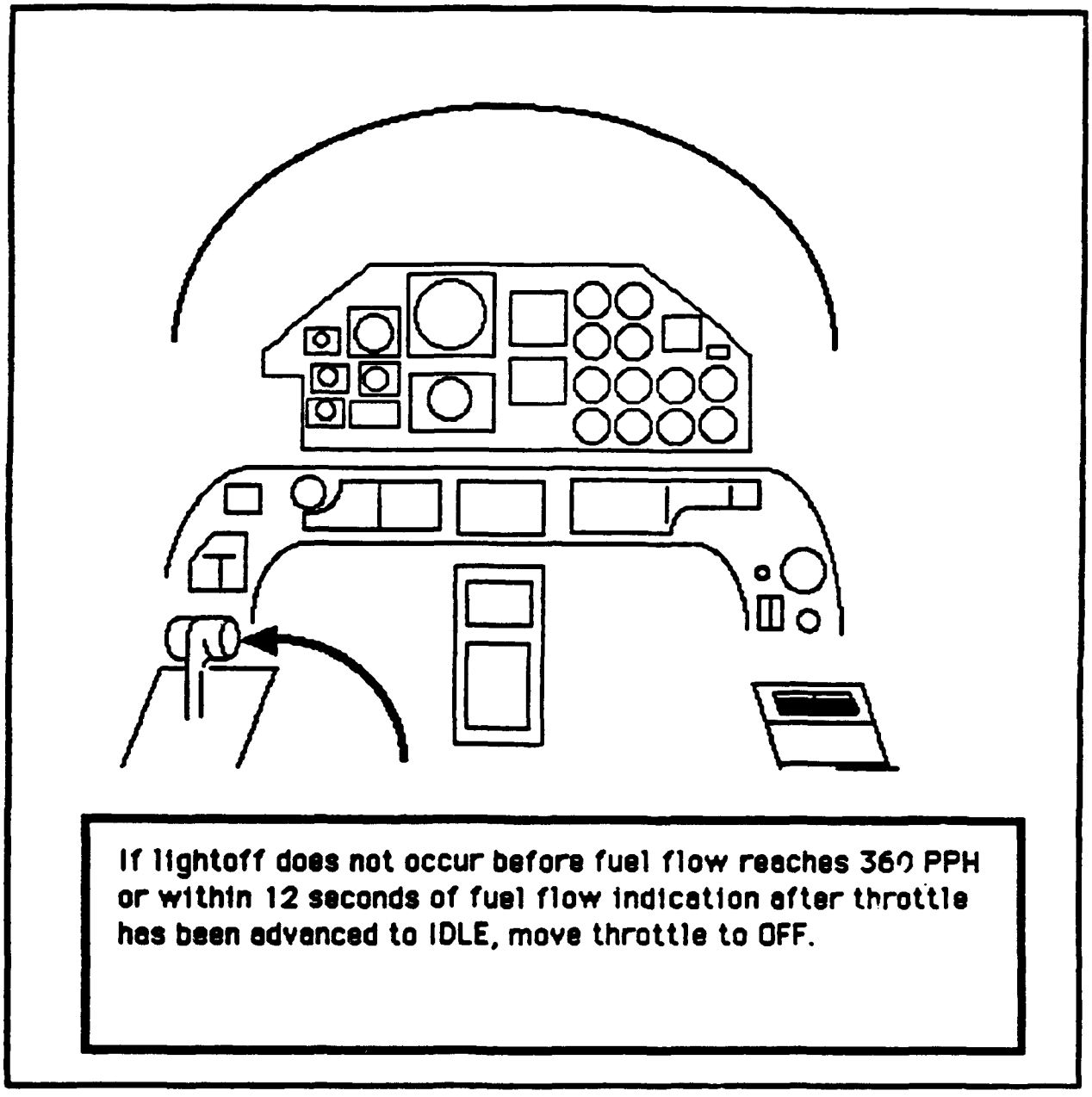


Figure 43. A Safety Rule Instructed

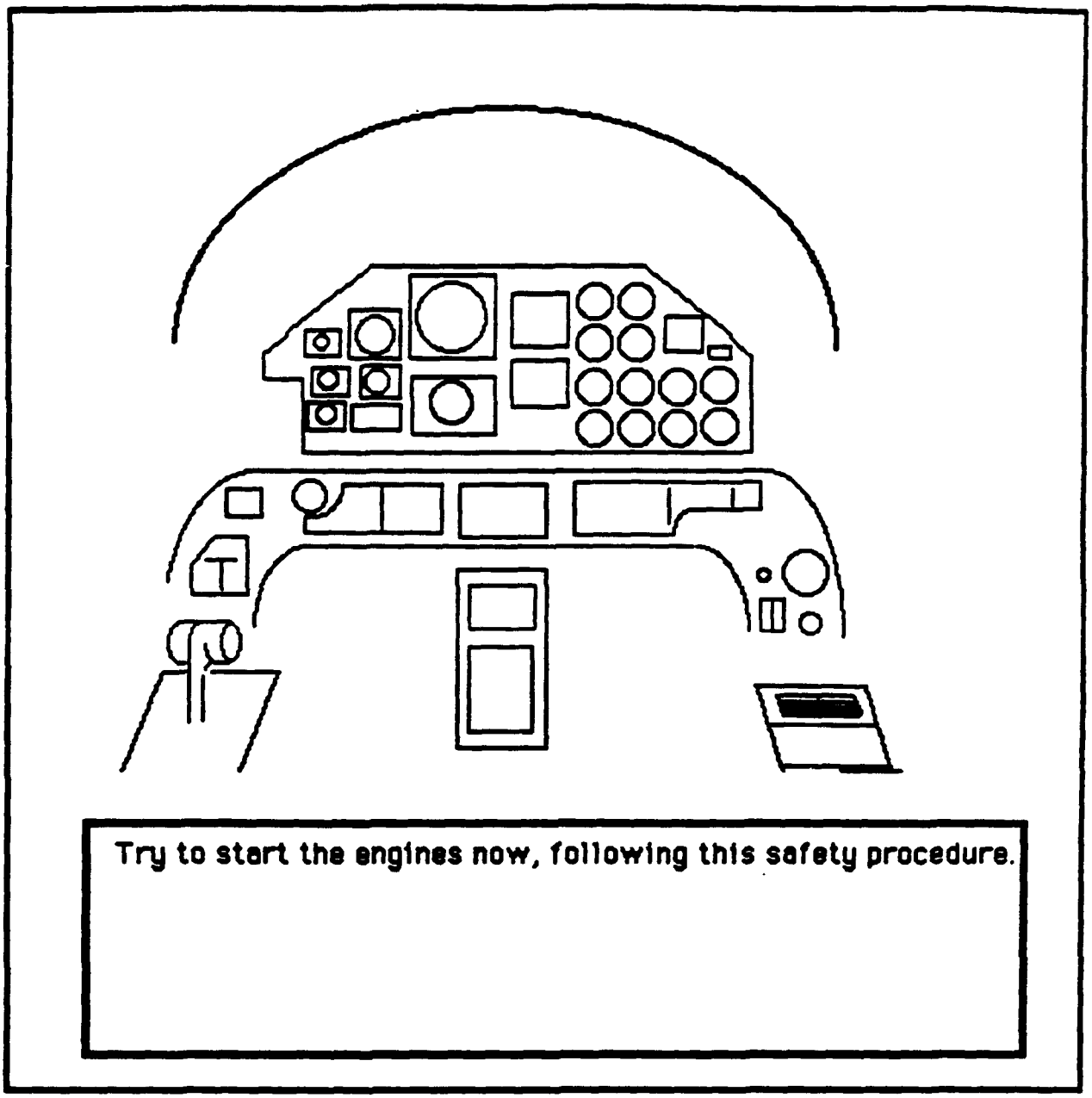


Figure 44. Exercise of a Safety Rule

If the learner commits errors during the process, RAPIDS automatically detects and corrects them. The student's performance, including requirements for remediation, is automatically recorded.

Instruction in Theory of Operation (Functional Equipment Model)

Figures 45 and 46 are examples of the instruction in internal functions.

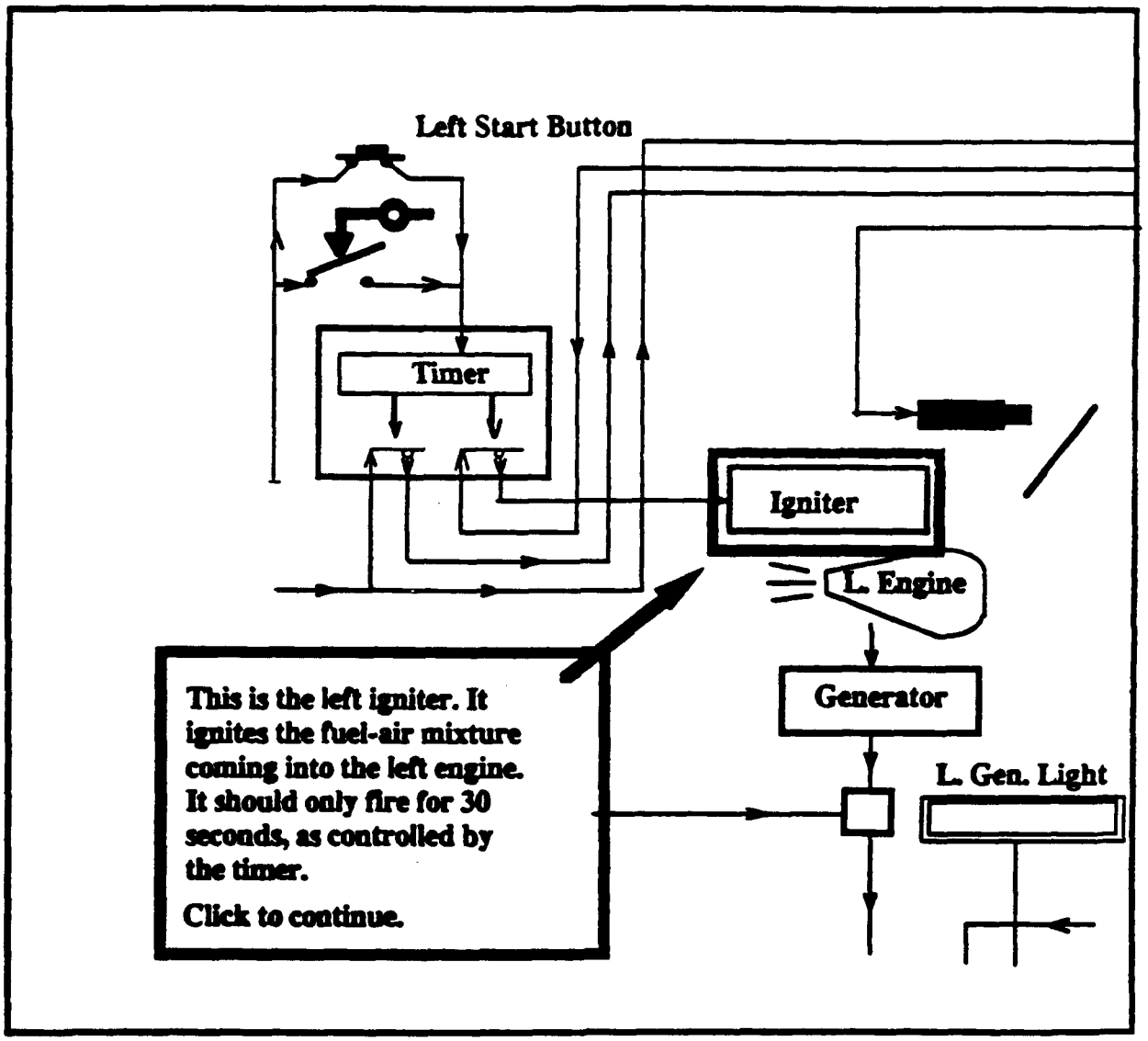


Figure 45. Instruction in Component Functions

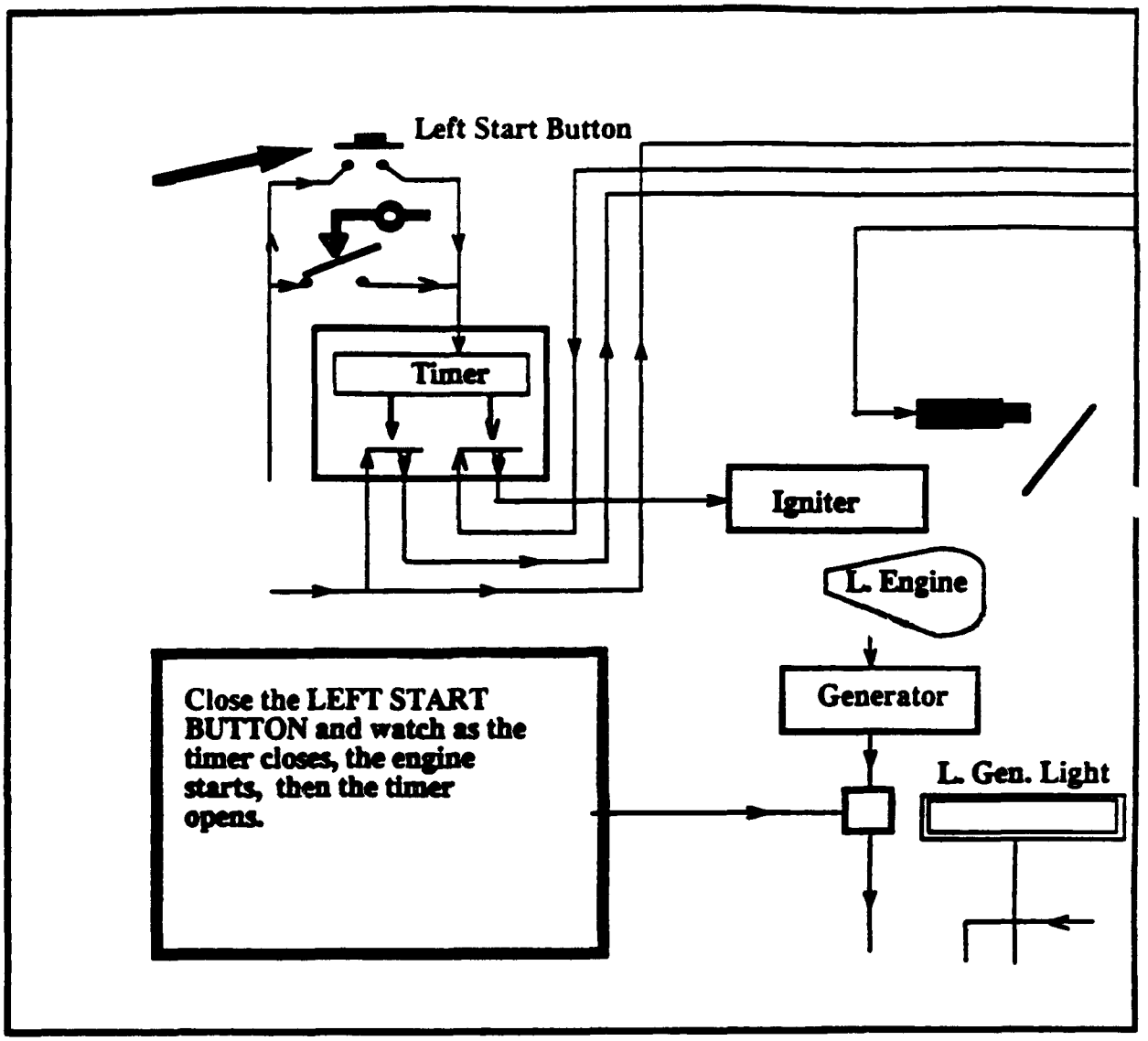


Figure 46. Instruction in Component Functions (continued)

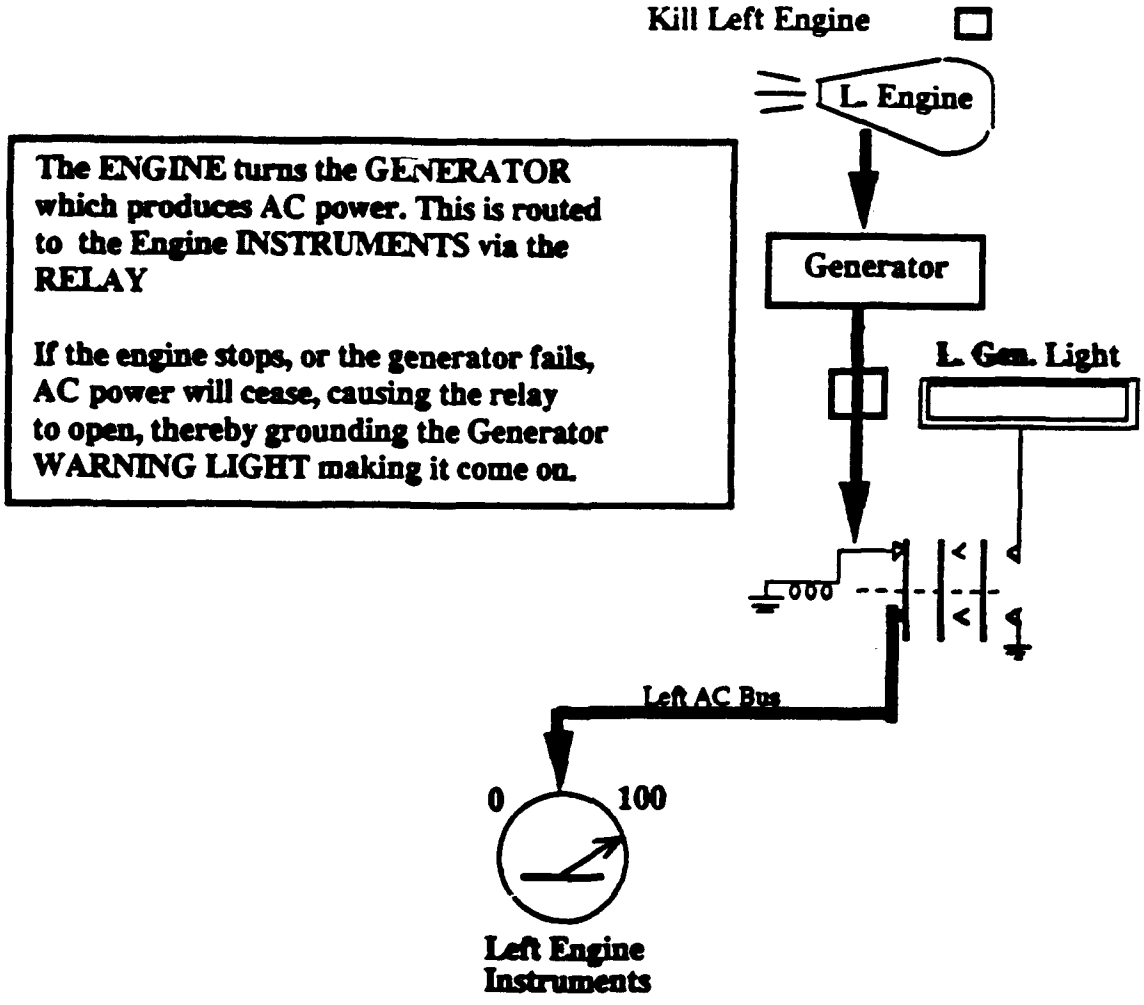


Figure 47. Instruction of AC Power Generation

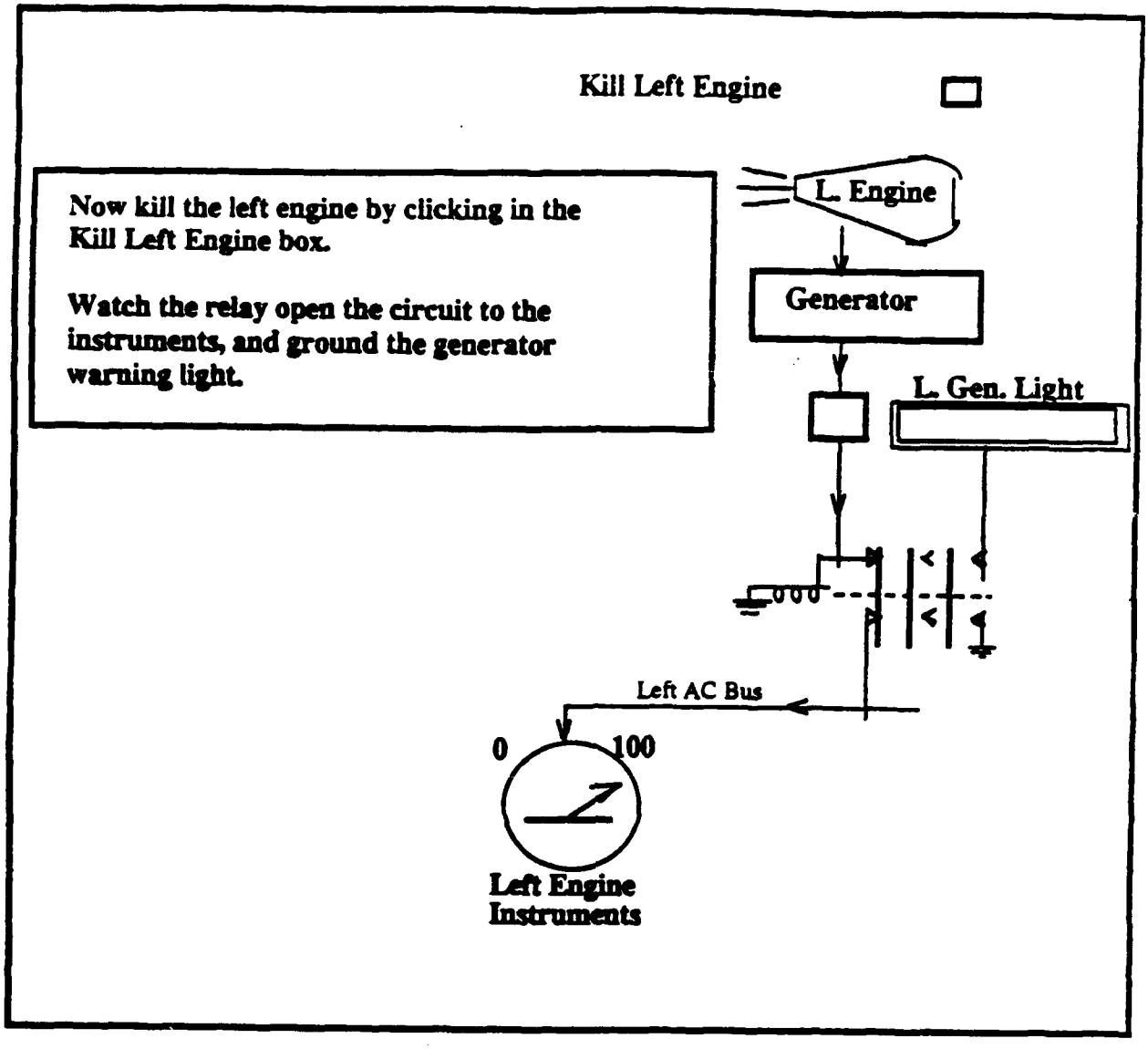


Figure 48. Instruction of AC Power Generation (continued)

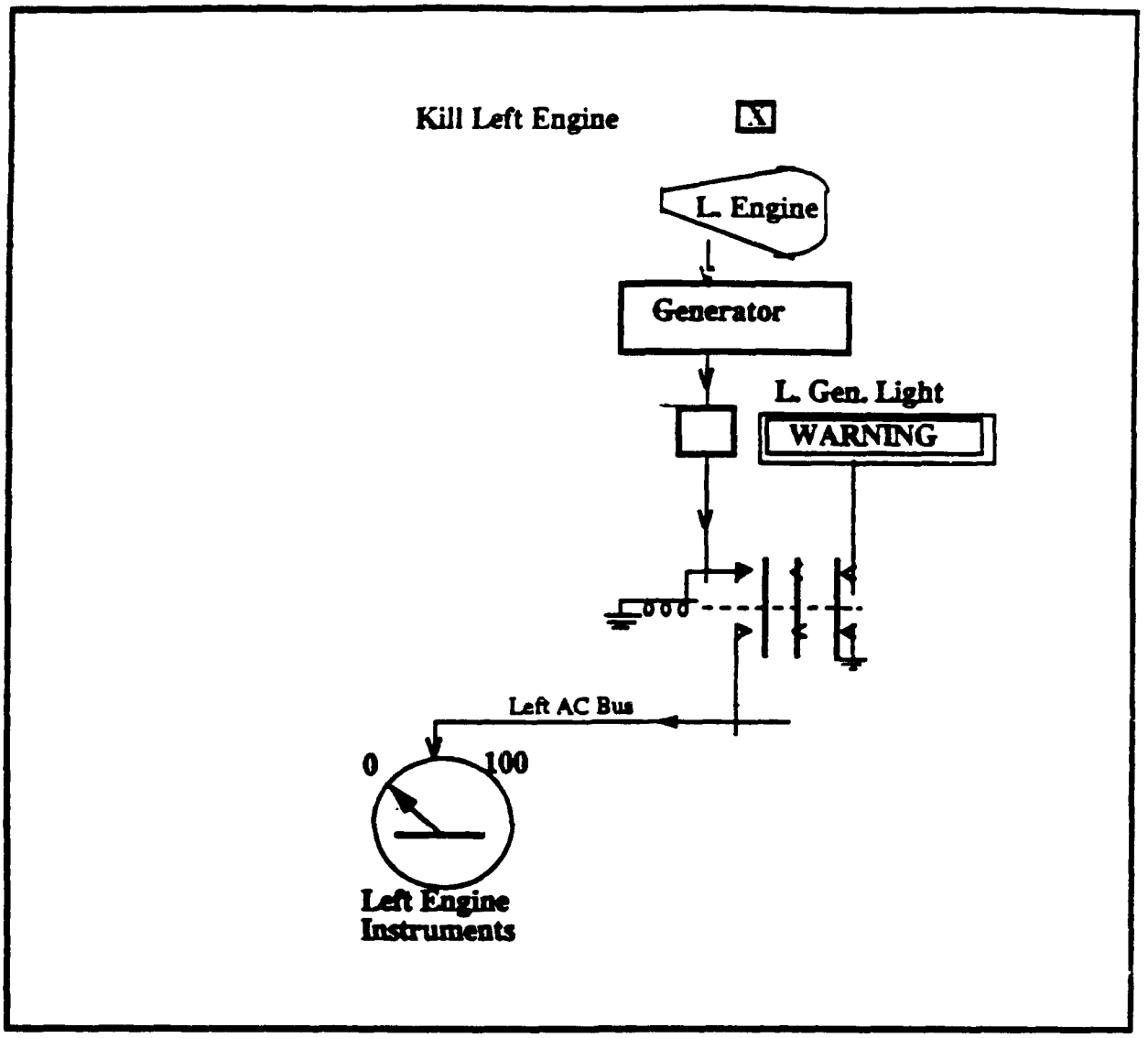


Figure 49. Instruction of AC Power Generation (continued)

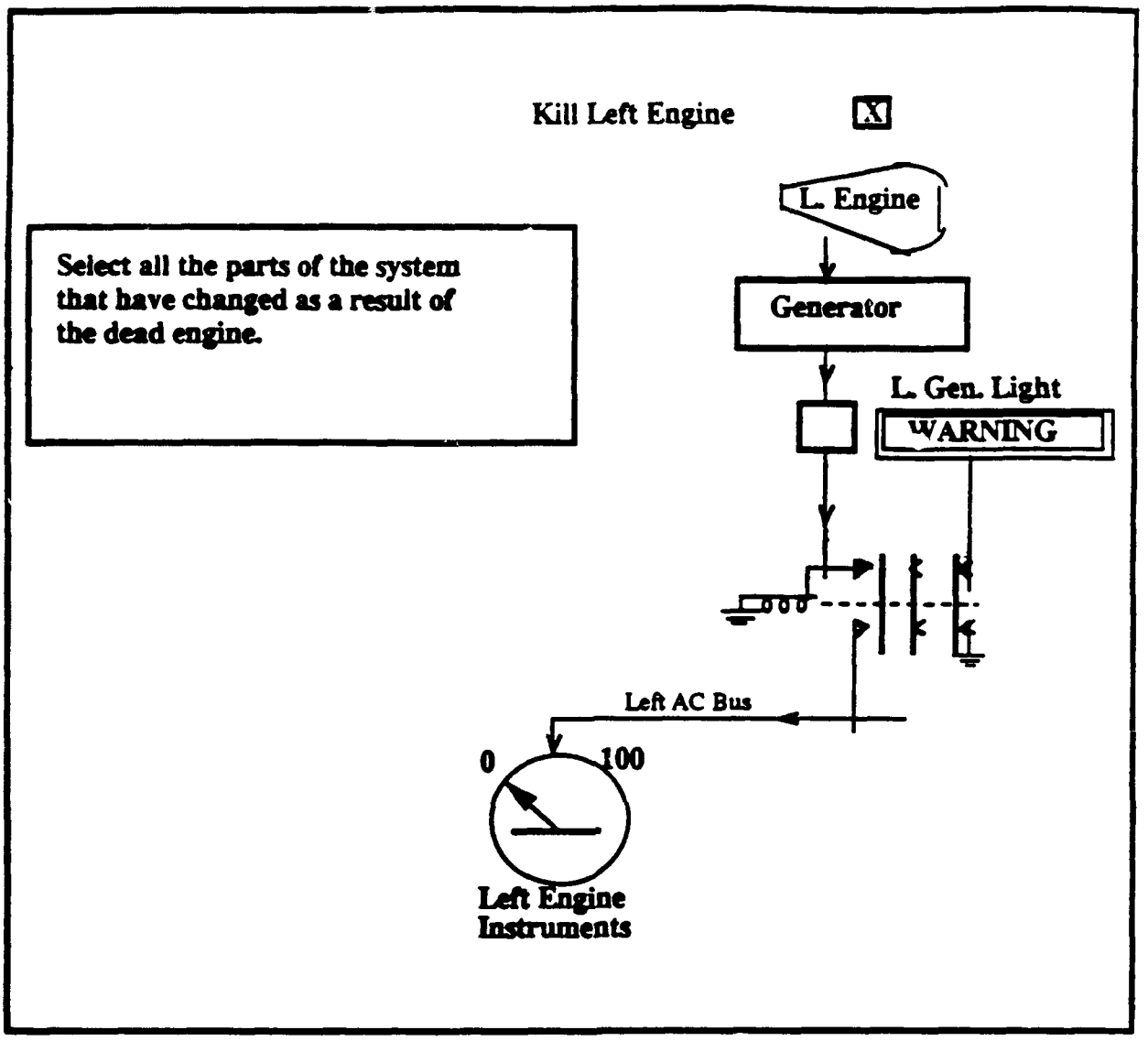


Figure 50. Instruction of AC Power Generation (continued)

Cross-over Theory

The function of the cross-over relay is explained and demonstrated in the following sequence.

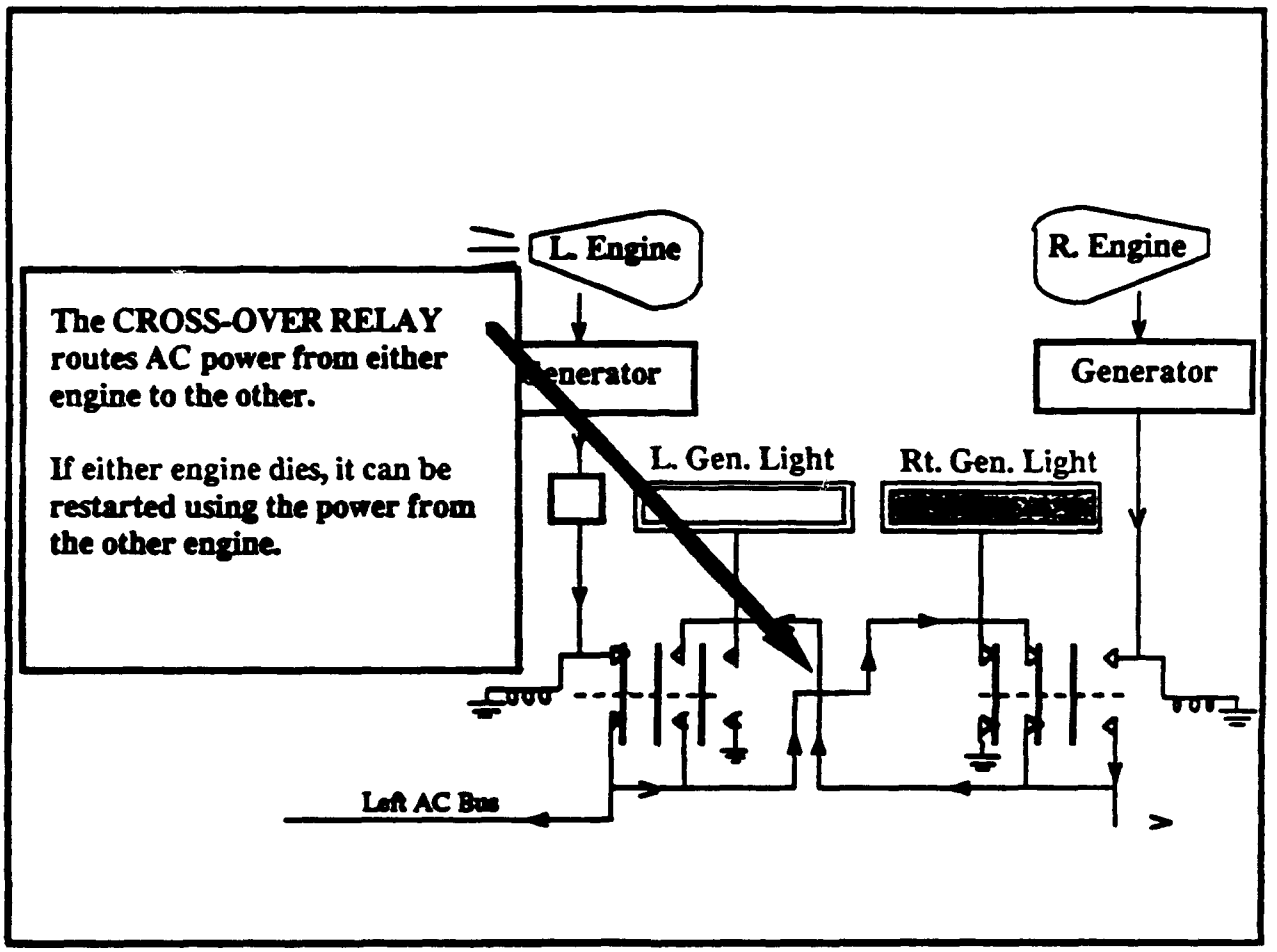


Figure 51. Function of the Cross-over Relay

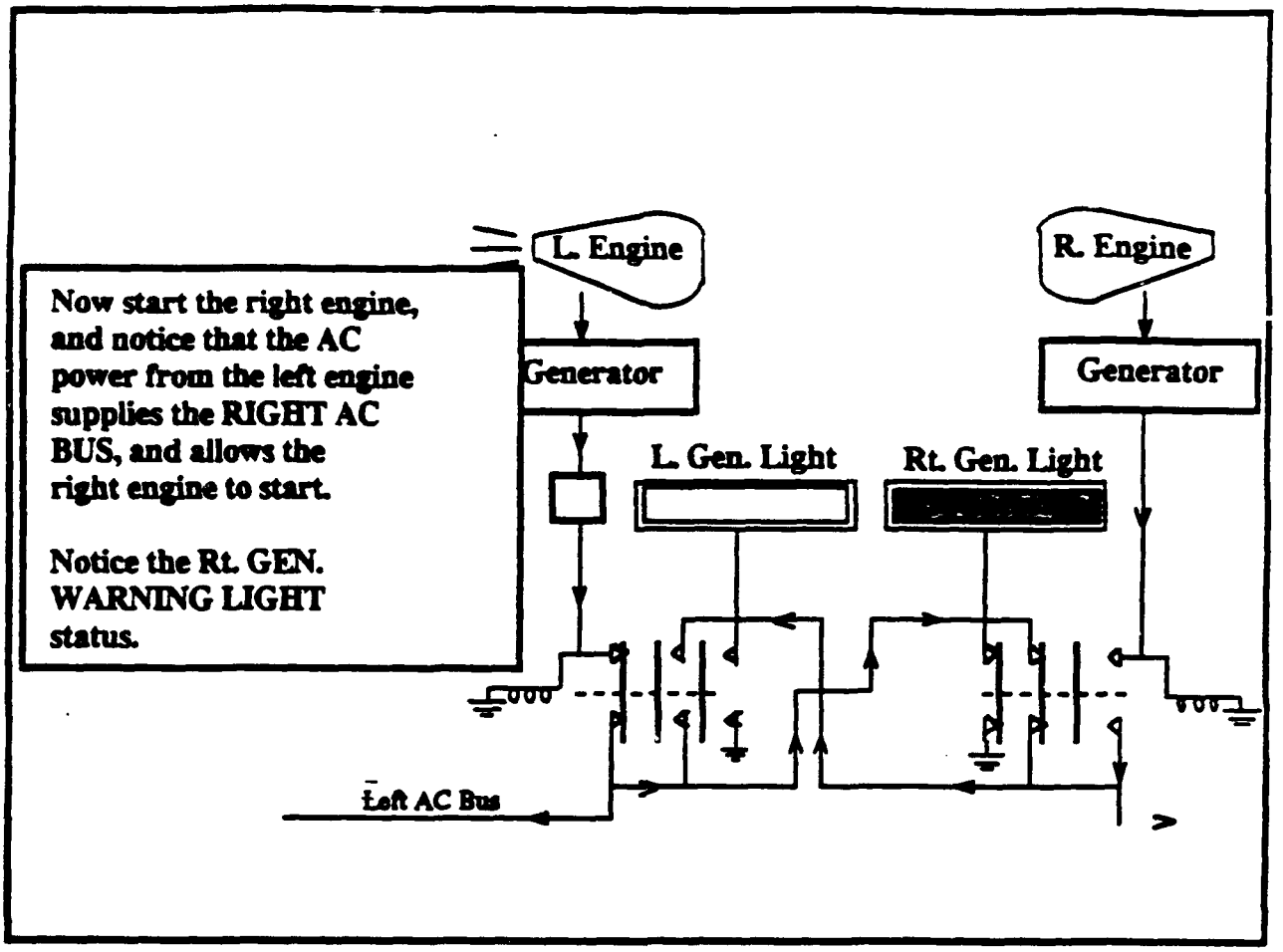


Figure 52. Function of the Cross-over Relay (continued)

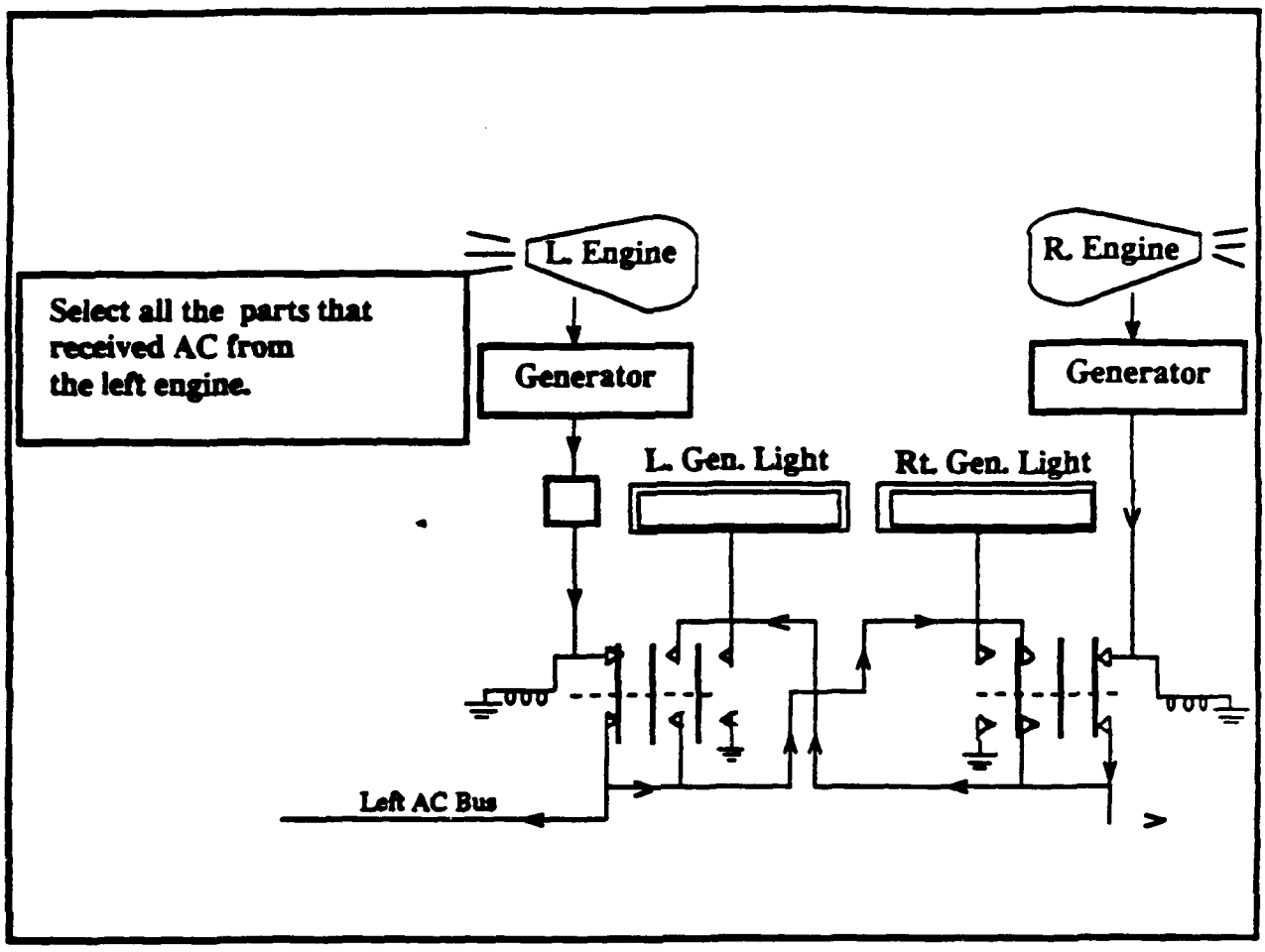


Figure 53. Function of the Cross-over Relay (continued)

Diagnostic Training

Test Evaluation

The learner is drilled in *detecting* abnormal indications.

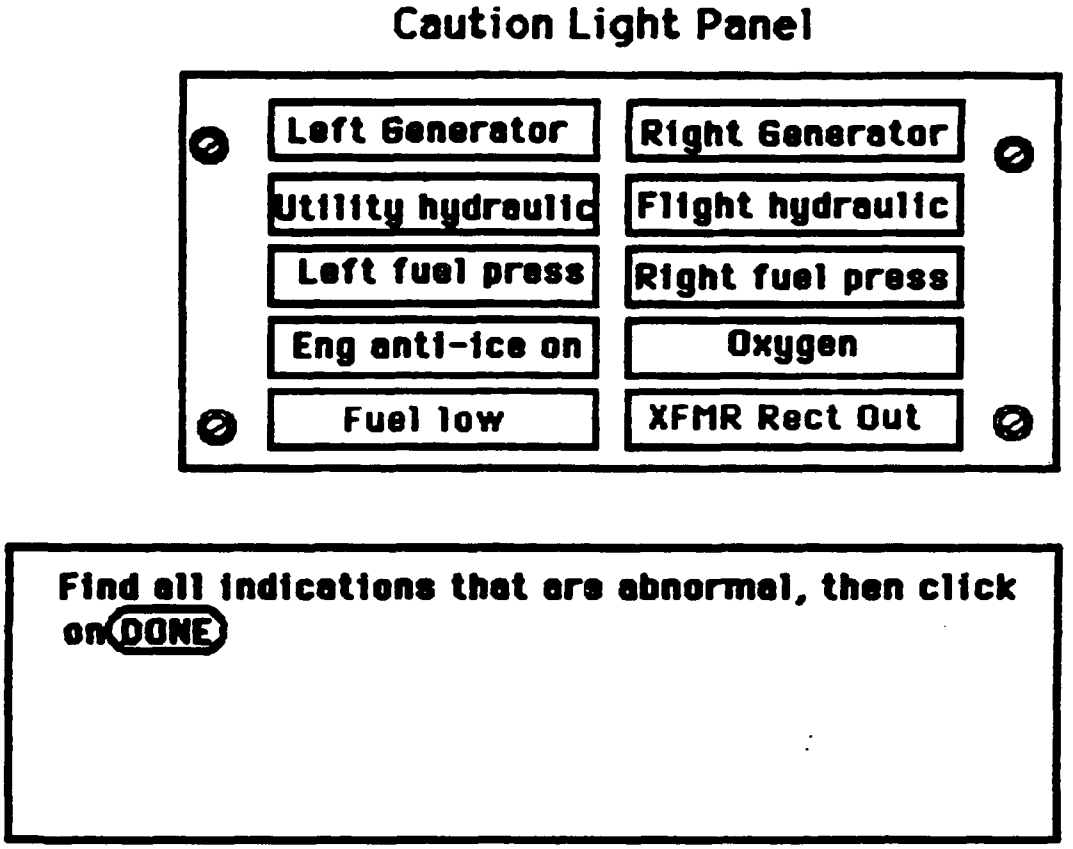


Figure 54. Drill in Detecting Abnormal Indications

Symptom Interpretation

The learner is drilled in identifying possible causes of abnormal readings.

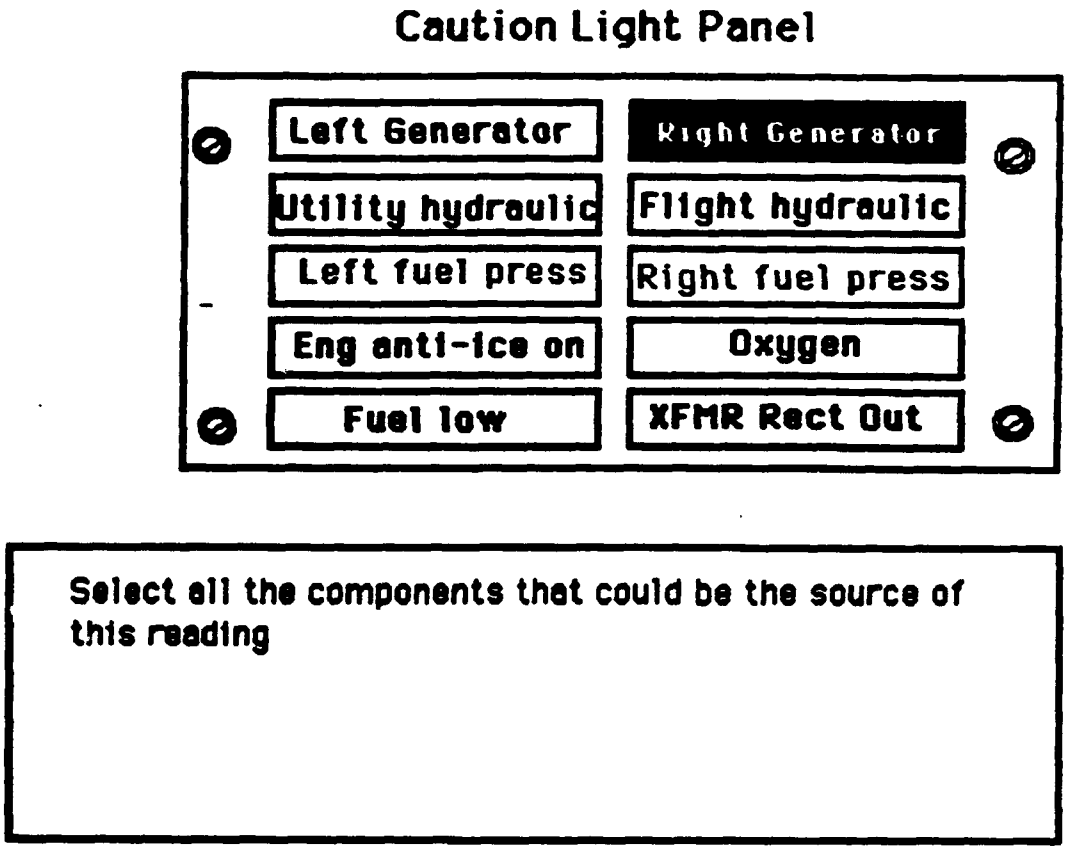


Figure 55. Drill in Identifying Possible Causes of Abnormalities

Fault Effects

RAPIDS drills the learner in identifying *what indicators would be affected* by various faults.

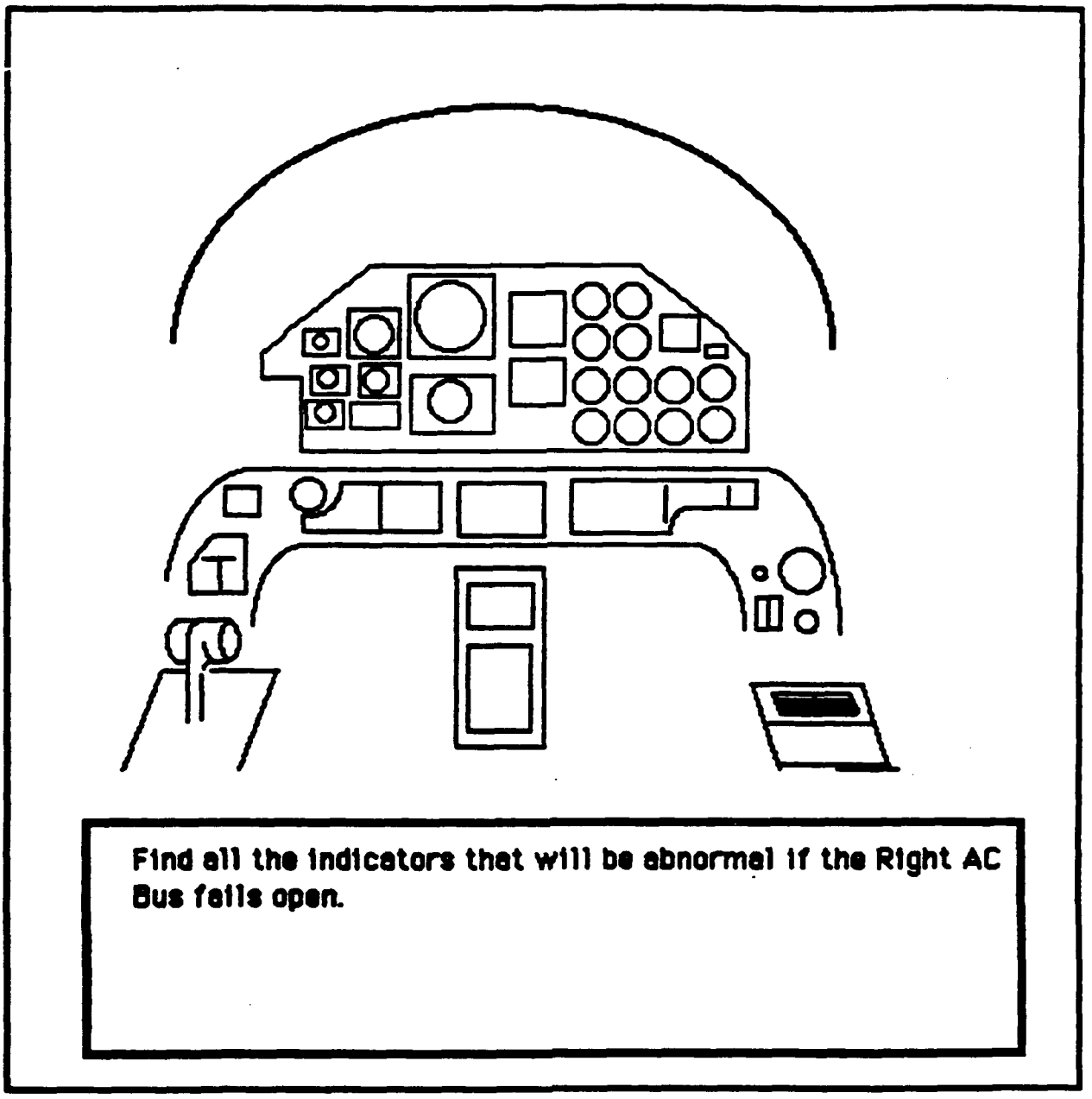


Figure 56. Fault-effect Instruction

The student selects all the parts that would be affected. As he or she does this, RAPIDS lists them and indicates if the selection is correct. Any errors are corrected, and omissions are automatically noted.

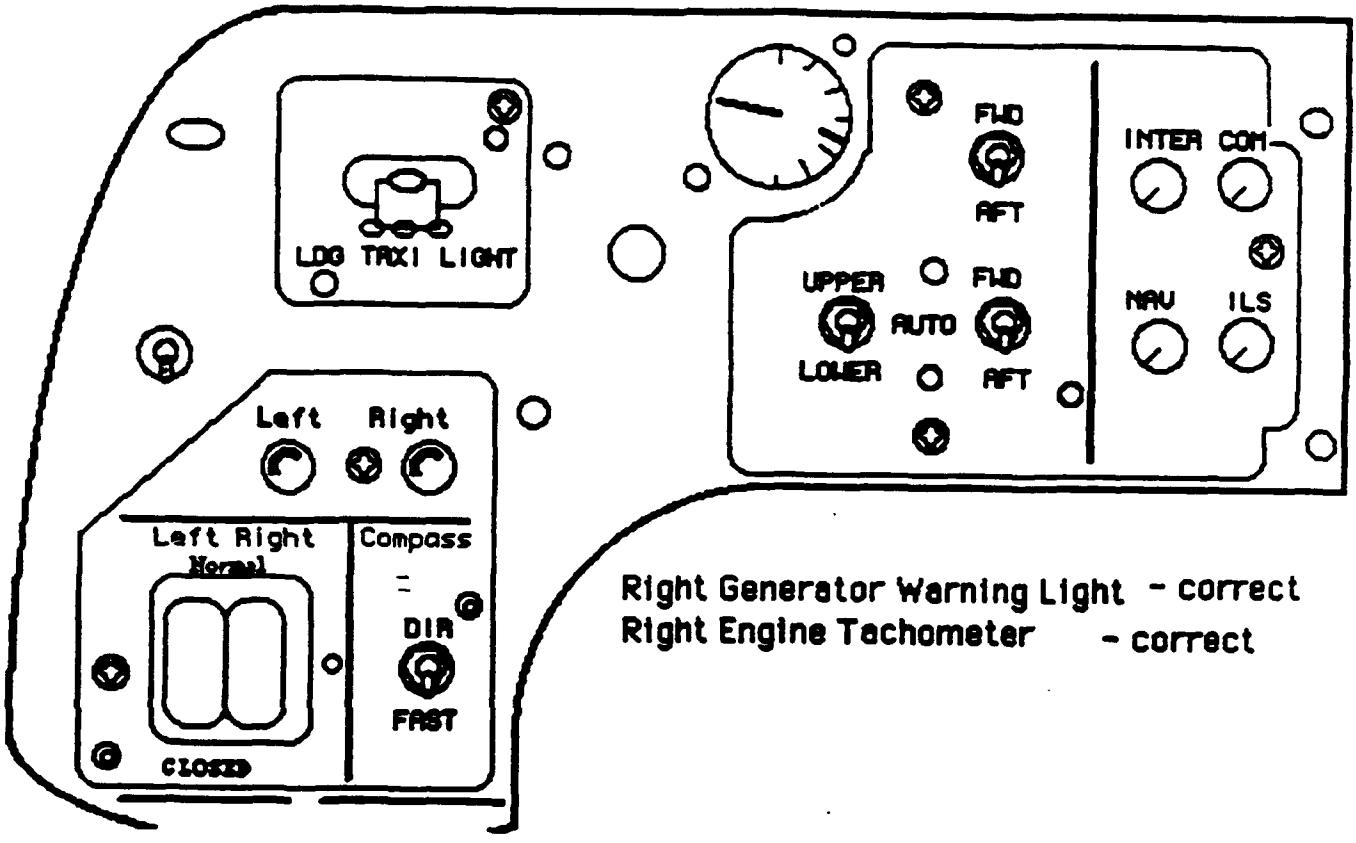


Figure 57. Fault-effect Instruction (continued)

Summary

In this example, the maximum instruction time is 47.75 hours; the average student might get through the instruction in half that time. This instruction was produced in about 70 total person-hours. Of this, 48 hours were devoted to developing the simulation, a fixed amount that is unrelated to the amount of instruction produced.

IV. CONCLUSIONS (Nason)

The Advanced Instructional Design Advisor is an R & D project being conducted by the Air Force Human Resources Laboratory in response to an Air Training Command (ATC) Manpower, Personnel, and Training Need calling for improved guidelines for authoring computer-based instruction (CBI) (MPTN 89-14T). The Advanced Instructional Design Advisor will provide subject matter experts who have no background in computer-based instructional systems with automated and intelligent assistance in the design and development of CBI. The goal is to reduce CBI development time while insuring that the instructional materials are effective.

The initial phase of this project established the conceptual framework and functional specifications for the Advanced Instructional Design Advisor, an automated and intelligent collection of tools to assist subject matter experts who have no special training in the design and development of effective computer-based instructional materials. This technical paper is part of the second phase of the project, which provided the design specifications for an experimental prototype.

A large portion of aircraft maintenance instruction concerns equipment-based troubleshooting or fault isolation. These consultants believed simulation capability is especially helpful in introducing troubleshooting to student. The consultants explored the possibility of integrating RAPIDS, an instructional system which provides computer-based simulation authoring and delivery capabilities with the Advanced Instructional Design Advisor. That exploration produced the two papers in this volume.

In section II, Drs. M. David Merrill and Mark K. Jones of M. David Merrill Consulting, described the integration of transaction shells (TRX), which are the basis of the Advanced Instructional Design Advisor, with RAPIDS. According to Merrill and Jones, RAPIDS could be used not only to build simulations but also to perform the knowledge analysis function. RAPIDS could help build a knowledge representation model, which is the product of the knowledge analysis function.

Knowledge in the model, produced when developing a simulation with RAPIDS, describes objects and their behavior. The Advanced Instructional Design Advisor recognizes the objects as entities and the behavior as processes. Entities and processes are two of the three types of "frames" which are linked into a network and used by the Advanced Instructional Design Advisor to create instruction. Thus, using the entities and processes produced by RAPIDS, the Advanced Instructional Design Advisor could automatically develop instruction. The Advanced

Instructional Design Advisor could also incorporate the simulations developed by RAPIDS into that the instructional material produced.

Dr. Douglas Towne, from the Behavioral Technology Laboratories at the University of Southern California, discussed the instructional development process using the RAPIDS system. When developing simulations with RAPIDS, instructional experts and subject matter experts determine the course structure and use software to represent graphically that structure within RAPIDS. They also determine the time and proficiency required of students.

To build the device simulation, a subject matter expert plans the simulation. He or she develops schematic functional models and produces graphic physical representations. The various states of these models are then specified along with the conditions producing them and the resulting effects. When the schematic and physical models are linked by the SME, the physical model becomes operational and can be altered by changing the states in the functional model.

RAPIDS automatically produces most of the fault isolation instruction associated with the equipment by linking the failed states to their effects and producing instruction routines. The SME then produces the remaining instruction required by using the graphic models to instruct nomenclature, function and procedures.

Towne's paper concludes by giving an example of the instruction a student would receive on the T-38 jet engine starter equipment. RAPIDS instructs and drills the student on equipment name and function. RAPIDS then instructs and drills procedures to start the jet engine, following safety rules. Finally, RAPIDS uses the schematic functional model and the physical model to instruct and drill the student on the operation of the equipment, including fault indications and isolation procedures.

Two significant conclusions resulted from this consideration of the need for a simulation authoring capability in AIDA. First, while there are many desirable features in RAPIDS and its successors, it would be too costly and time-consuming to incorporate all of these features in the third phase experimental prototype (XAIDA). Second, because XAIDA does require some simulation creation capability, there will be a simpler, low-order simulation authoring capability created to satisfy near-term needs.

REFERENCES

Hickey, A. E., Spector, J. M., & Muraida, D. J. (1991). Design Specifications for the Advanced Instructional Design Advisor (AIDA) (Volume 1 of 2) (AL-TR-1991-0085-Vol 1). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory.

Hickey, A. E., Spector, J. M., & Muraida, D. J. (1991). Design Specifications for the Advanced Instructional Design Advisor (AIDA) (Volume 2 of 2) (AL-TR-1991-0085-Vol 2). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory.

Jones, M. K. Li, Z. & Merrill, M. D. (1990). Domain knowledge representation for instructional analysis. Educational Technology, 30 (10), 7-32.

Jones, M. K. Merrill, M. D., & Li, Z. (1990). Implementation of an expert system for instructional design: Design of the strategy analysis component. McLean, VA: Human Technology Inc.

Spector, J. M. (1990). Designing and Developing an Advanced Instructional Design Advisor (AFHRL-TP-90-52). Brooks AFB, TX: Training Systems Division, Air Force Human Resources Laboratory.